International Handbooks on Information Systems

Titles in the Series

P. Bernus, K. Mertins and G. Schmidt (Eds.)
Handbook on Architectures of Information Systems
ISBN 3-540-64453-9

M. Shaw, R. Blanning, T. Strader and A. Whinston (Eds.)
Handbook on Electronic Commerce
ISBN 3-540-65822-X

J. Błażewicz, K. Ecker, B. Plateau and D. Trystram (Eds.)
Handbook on Parallel and Distributed Processing
ISBN 3-540-66441-6

H. H. Adelsberger, B. Collis and J. M. Pawlowski (Eds.)
Handbook on Information Technologies for Education and Training
ISBN 3-540-67803-4

C. W. Holsapple (Ed.)
Handbook on Knowledge Management 1
Knowledge Matters
ISBN 3-540-43527-1
Handbook on Knowledge Management 2
Knowledge Matters
ISBN 3-540-43527-1

J. Błażewicz, W. Kubiak, T. Morzy and M. Rusinkiewicz (Eds.)
Handbook on Data Management in Information Systems
ISBN 3-540-43893-9

S. Staab and R. Studer (Eds.)
Handbook on Ontologies
ISBN 3-540-40834-7

S. O. Kimbrough and D. J. Wu (Eds.)
Formal Modelling in Electronic Commerce
ISBN 3-540-21431-3

Steven O. Kimbrough
D. J. Wu
Editors

Formal Modelling in Electronic Commerce

With 83 Figures
and 60 Tables

Springer

Professor Steven O. Kimbrough
University of Pennsylvania
Operations & Information Management
565 Jon M. Huntsman Hall
3730 Walnut Street
Philadelphia, PA 19104-6340, USA
E-mail: kimbrough@wharton.upenn.edu

Professor D. J. Wu
800 West Peachtree Street, NW
College of Management
Georgia Institute of Technology
Atlanta, GA 30332-0520, USA
E-mail: dj.wu@mgt.gatech.edu

Cataloging-in-Publication Data applied for

Library of Congress Control Number: 2004114081

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data available in the internet at *http://dnb.ddb.de*

ISBN 3-540-21431-3 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springeronline.com

Printed in Germany

Cover design: Erich Kirchner, Heidelberg
Production: Helmut Petri
Printing: Krips

SPIN 10997895 Printed on acid-free paper – 42/3130 – 5 4 3 2 1 0

Preface

Questions abound.

- What is the deep structure of a purchase order, or for that matter any other business document, such as an invoice, bill of lading, receiving report, or contract? How can a purchase order (or other document) be represented so that organizations may—automatically, without direct human involvement—create, send, reason correctly about, and act properly on the purchase order (or other document)?
- How may a purchase order (or other business document) be structured without ad hoc-ness, so that it may be used by automated agents everywhere?
- Beyond sending simple messages, how can the need of organizations to conduct extended conversations automatically be supported?
- Is it possible, and if so how, for artificial agents to negotiate contracts and trade procedures? What are the possibilities? the limitations?
- What are the underlying principles of reasoning about such subtle concepts as obligations, permissions, actions, and norms? How might these principles be formalized and employed generally in electronic commerce automation?
- How can artificial agents discover and learn strategies that are effective in commercial—hence, game-theoretic—contexts? What are the candidate learning regimes and what are their performance characteristics? Will they be exploitable? Will they be exploiting?
- What are the behavioral characteristics of markets and other commercial transaction spaces when populated by artificial agents that are intelligent, adaptive, and learning? Will they be stable? Will they be efficient?

These questions, and many other questions like them, matter because important prospective advances in electronic commerce—the progressive automation of the conduct of business—depend on answering them. The questions are deep. They merit and will receive enduring attention. It is widely agreed that the state-of-the-art on these questions calls for intensive and protracted attention. Happily, the issues they raise admit of imperfect solutions, which may profitably be translated into practice, then replaced as knowledge improves. The scene is dynamic and filled with opportunities, both for fundamental contributions and for transfer of knowledge to practice.

The FMEC (Formal Modeling for Electronic Commerce) community is an informal, diverse, international, and very open group of researchers, who share an interest in these questions. The community has been operating actively since 1987, and has produced nearly a score of workshops (with attendant papers), nine special issues of journals (totaling 59 papers), and at least 21 Ph.D. theses. In addition, several score papers have appeared in journals,

conferences, and workshops. The community has been fecund in its ideas and prolific in its published output.

The purposes of this volume are several:

- To present the latest work from the FMEC community
  The book contains 14 new papers, representing the best of the current FMEC work.
- To present FMEC papers of historical and conceptual import, which are not otherwise available in the open literature
  The book contains 5 such papers.
- To provide an historical and, mainly, conceptual guide to the issues that have been addressed by the FMEC community
  The book's first chapter provides an integrating essay on FMEC work. It will be a useful guide to the reader of this book and to the researcher who wishes to probe further. In addition, the book contains a unified master bibliography of several hundred entries, as well as a comprehensive index.

We hope, and believe, this volume will be useful for researchers and graduate students with interests in electronic commerce or in the fundamental issues engaged by the questions listed above. Practitioners, especially those involved in research or development of leading-edge systems, will also find here much to reckon with. Most, if not all, of the ideas explored here are ripe for conversion to applications.

We end with a note of thanks. Since 1987, dozens of people have made important contributions to the creation and maintenance of the FMEC community. We thank them all, even if it is not possible to name them all. Special thanks, however, go to four senior individuals who have been exemplary in the support and encouragement they have given this community.

- Melvin F. Shakun, editor of *Group Decision and Negotiation*
- Ralph H. Spague, Jr., organizer of the Hawaii International Conference on System Sciences
- Andrew Whinston, editor of *Decision Support Systems*
- Vladimir Zwass, editor of *Journal of Management Information Systems* and *International Journal of Electronic Commerce*

Finally, thanks to Mike Shaw for suggesting this project and to the editors at Springer for their patience.

Bala Cynwyd, PA and Atlanta, GA — *Steven O. Kimbrough*
September 2004 — *D.J. Wu*

Contents

Preface V

FMEC: Overview and Interpretation 1
Steven O. Kimbrough, D.J. Wu
1 Introduction 1
2 Formalisms 2
3 Themes 5
4 Topics 6
5 A Brief Guide to the Volume 7
6 Upwards and Onwards 13
7 Acknowledgements 18
A FMEC Bibliographic History 18

Part I. Representation: Objects, Processes & Policies

Practical Contract Storage, Checking, and Enforcement for Business Process Automation 33
Alan Abrahams, David Eyers, Jean Bacon
1 Introduction 33
2 Contributions 35
3 Application Scenario 36
4 Overview 37
5 Occurrences 40
6 Kimbrough's Disquotation Theory 45
7 An Implementation of Kimbrough's Disquotation Theory 47
8 Contract Provision Monitoring 59
9 Contract Performance and Enforcement 61
10 Software Implementation 65
11 Related Work 71
12 Conclusions 72
13 Acknowledgements 73
A Coverage Checking Rules 74

Legitimacy Checking in Communicative Workflow Design 79
Aldo de Moor, Hans Weigand
1 Introduction 79
2 The Extended Workflow Loop 81
3 Extended Workflow Loop Norms 82
4 Workflow Loop Schemas 86
5 A Method for Legitimacy Checking 89

6 Conclusions 97

CANDID Specification of Commercial and Financial Contracts, Part I: Syntax & Formal Semantics of CANDID 101
Ronald M. Lee
1 Introduction 101
2 The Language L_1 103
3 Re-Interpretation of Predicates 106
4 Many-Sorted, Type-Theoretic Languages 107
5 λ Abstraction 108
6 Operations, Definite Reference 110
7 Summary of the Language L_v 111
8 Character Strings, Labels 114
9 Numbers and Measurement 115
10 Time, Realization, Change 117
11 Possible Worlds, Intensions 122
12 Summary of the Language IL 125
13 Action 130
14 Modals, Deontic Operators 131
15 Summary of the Language CANDID 136

CANDID Specification of Commercial and Financial Contracts, Part II: Formal Description of Economics Actors and Objects 145
Ronald M. Lee
1 Introduction 145
2 Economic Actors 150
3 Economic Objects 153
4 Summary 157

CANDID Specification of Commercial and Financial Contracts, Part III: CANDID Specification of Financial Concepts 159
Ronald M. Lee
1 Introduction 159
2 Additional Definitions, Notational Conventions 160
3 Elementary Financial Concepts 161
4 Financial Instruments 167
5 Concluding Remarks 176

Performatives, Performatives Everywhere but Not a Drop of Ink 177
Ronald M. Lee
1 Performative Aspects of Commerce and Public Administration 177
2 Issues for Open Electronic Commerce 186
3 From Ink to Bits: Original, Signed Writings 188
4 Computational Modeling of Documentary Procedures 192
5 Protocols for Procedure Adoption 194

6 Discussion and Further Research Directions 197
7 Acknowledgments 198

EDI, XML, and the Transparency Problem in Electronic Commerce 201
Steven O. Kimbrough
1 EDI and the Transparency Problem 202
2 XML's Pertinent Virtues and Limitations 203
3 Communications Requisites 206
4 microFLBC and the Transparency Problem 211
5 Back to XML 221
6 Conclusion 224

Part II. Applications

Designing Control Mechanisms for Value Exchanges in Network Organisations 231
Vera Kartseva, Yao-Hua Tan
1 Introduction 231
2 A Methodology for Designing Control Mechanisms 233
3 Modelling Business Value Models 234
4 Modelling Sub-Ideal Situations 236
5 Distinguishing Different Types of Control Mechanisms 243
6 Conclusions 244

Sim-I-Space: An Agent-Based Modelling Approach to Knowledge Management Processes 247
Max Boisot, Ian MacMillan, Kyeong Seok Han, Casey Tan, Si Hyung Eun
1 Introduction 247
2 Model Architecture 248
3 Model Components 252
4 Acknowledgements 263
A Description of Variables 263
B Detailed Model Specification with Example 278

Part III. Communication

On Representing Special Languages with FLBC: Message Markers and Reference Fixing in SeaSpeak 297
Steven O. Kimbrough, Yinghui (Catherine) Yang
1 Introduction 297
2 Special Languages 298
3 Two Problems 300

4 Background on SeaSpeak 302
5 Prototype Example: INFORMATION 304
6 Problems of Reference Fixing 307
7 Formalizing Distributed Descriptions into FLBC 312
8 Analysis of the Remaining SeaSpeak Message Markers 315
9 Discussion and Conclusion 320

A Note on Modelling Speech Acts as Signalling Conventions . . 325
Andrew J.I. Jones, Steven Orla Kimbrough
1 Introduction 325
2 Asserting: Two Prototypes 326
3 Other Speech Acts 330
4 Discussion: Towards Deployment 334
5 Summary and Conclusion 339

Dynamic Conversation Structures: An Extended Example 343
Scott A. Moore
1 Introduction 343
2 Extended Example 345
3 Conclusion 359

Part IV. Agents and Strategic Interactions

Investigating the Value of Information and Computational Capabilities by Applying Genetic Programming to Supply Chain Management 363
Scott A. Moore, Kurt Demaagd
1 Introduction 363
2 The Evolutionary Process 366
3 Plan of Investigation 380
4 Summary 387
A Terminals and Functions 388
B Settings for a Scenario 389
C Computing-Related Information 390

Multi-Agent Simulation of Financial Markets 393
Olga Streltchenko, Yelena Yesha, Timothy Finin
1 Introduction 393
2 Automation of Modern Financial Markets 394
3 Simulation of Financial Markets 399
4 A Multi-Agent Environment for Financial Market Simulation 404
5 Conclusions 416

Adaptive Agents in Coalition Formation Games 421
Alex K. Chavez
1 Introduction 421
2 Background and Description 422
3 Learning Models 425
4 Results 430
5 Conclusion 435
6 Acknowledgements 437
A Parameter Estimates 438

On Learning Negotiation Strategies by Artificial Adaptive Agents in Environments of Incomplete Information 445
Jim R. Oliver
1 Introduction 445
2 Overview of the Approach 448
3 AAA Platform 448
4 Structure of the Bargaining Space 451
5 Experimental Testing and Results 453
6 Conclusion 459

A Note on Strategic Learning in Policy Space 463
Steven O. Kimbrough, Ming Lu, Ann Kuo
1 Introduction 463
2 Background: Games and Decisions 463
3 Repeated Games 466
4 Simple Reinforcement Learning 467
5 Learning in Policy Space 469
6 Discussion 473
7 Acknowledgements 474

Learning and Tacit Collusion by Artificial Agents in Cournot Duopoly Games 477
Steven O. Kimbrough, Ming Lu, Frederic Murphy
1 Introduction 477
2 The Duopoly Game: Holt's Cournot Model 478
3 Background and Application Context: Electricity Markets 479
4 Framework for Agent Learning 481
5 Molecular Strategies in the Cournot Game 483
6 Summary of Results 486
7 Discussion 487
8 Acknowledgements 490

A Note on Working Memory in Agent Learning 493
Fang Zhong
1 Introduction 493
2 The Exchange Game 494
3 Learning Mechanism 497
4 Experiments on Working Memory 499
5 Discussion 504
6 Conclusion 505
7 Acknowledgements 506
A Parameter Values 506

Investigations of Granularity and Payoffs in 2×2 Games under Replicator Dynamics 509
Sofia Chajadine, Daniel Mack, Aaron Jeffrey Slan
1 Introduction 509
2 The Games 510
3 Methodology 514
4 Discussion of Findings 516
5 Acknowledgements 525
A Summary of Simulations 525

Part V. References and Index

References 531

Index 557

FMEC: Overview and Interpretation

Steven O. Kimbrough[1] and D.J. Wu[2]

[1] University of Pennsylvania, Philadelphia, PA, USA,
kimbrough@wharton.upenn.edu
[2] Georgia Institute of Technology, Atlanta, GA, USA,
dj.wu@mgt.gatech.edu

Abstract. This paper is an integrative essay on the activities and intellectual concerns of the FMEC community. The paper frames these concerns around three 'non-standard' formalisms (logic, graphs, and procedures), three themes or general problems (representation, inference, and learning), and seven more specific topics (electronic data interchange, electronic contracting, speech acts, special logics, system and process modelling, strategy formation, and computational discovery). In addition, the paper introduces each of the chapters in this book and places them within the general FMEC framework. An appendix to the paper records the bibliographic history of the FMEC community.

1 Introduction

Electronic commerce is an attractive field for anyone interested in originating fresh ideas or in innovatively translating them into practice. Underlying technical progress in computing and communications continues a torrid advance. In consequence the deployment, full exploitation, and even conception of applications enabled by technical progress inevitably lag and go unrealized, at least for a time, sometimes a considerable time. Thus is created a permanent (if moving) frontier, open to insight and rewarding it.

Electronic commerce is equally attractive to those with a taste for fundamental challenges and with aspiration to make foundational contributions. The general challenge is "to expand the realm of the automated in a principled and generalized fashion."[1] To take up this challenge is to be confronted with any number of fundamental problems, calling for fundamental insight and innovation.

The papers in this volume address both kinds of challenges—e-commerce application challenges and e-commerce fundamental challenges—in detail. It will be useful, for the sake of interpreting these papers, to frame what it is that the FMEC community has been about. For that purpose, three facets or perspectives will serve to characterize, at least roughly, the intellectual concerns that have drawn the attention of researchers in the FMEC community:

[1] Since the early 1980s, this has been Steven O. Kimbrough's slogan for characterizing the mandate of Information Systems as a discipline.

- Formalisms
- Themes
- Topics

2 Formalisms

Formal modelling requires some formalism or other in which to express models. Moreover, electronic commerce is a capacious area of research and much formal modelling has long been directed at it. What distinguishes FMEC? How is it different from standard modelling work, say, in economics, or the management sciences? The FMEC community may perhaps best (but always approximately) be described as having focused on *non-standard* formal modelling for electronic commerce. What we may call the *standard modelling formalism* employs broadly algebraic (or equational) models of various sorts. The literatures of economics, the management sciences, and the various business disciplines (e.g., marketing, finance) are suffused with models using the standard formalism(s). Moreover, modelling in this style has been present and welcomed in the FMEC literature.[2]

From the outset, however, the FMEC community has been interested in problems and topics for which other formalisms are most naturally used. These other, 'non-standard' formalisms have been of broadly three kinds:

- Logic
- Graphs
- Procedures

Logic. The origin of the FMEC community was a series of Logic Modelling minitracks at the HICSS meetings (Hawaii International Conference on System Sciences), beginning in January 1987 (see §A, below). From the outset, and continuing to the present, a large segment of the FMEC community has focused on the use of formal logic as a modelling tool. The opening article in the first FMEC special issue, "Logic Modeling: A Tool for Management Science" [KL88], presents the case. In a nutshell, the promise of logic modelling is that:

1. Logic models are natural representational formalisms for any target system that is propositional, such as documents and messages used to conduct business.
2. Logic models afford construction of clear and rigorous models; and they bring with them the considerable foundational underpinnings of formal logic.
3. Via the logic programming paradigm, logic models are readily implemented and translated into applications. In this context, we may think of a logic model as an 'executable theory'.

[2] E.g., [Jer88,BCK00,Wu97]

Close to half of the FMEC papers appearing either in this volume or in the ten previous special issues of journals (see §A.2, below) employ a logic formalism.

Graphs. Graphs—articulated structures of nodes and arcs—are an especially apt formalism for electronic commerce.

1. Graphs are natural representational formalisms for processes (e.g., trade procedures, workflow processes) and indeed for systems. UML and all other formalisms used in systems analysis and design are largely graph-based. Networks—social, trading, transportation, etc.—appear regularly in e-commerce contexts, and networks are special kinds of graphs.
2. Graphs afford construction of clear and rigorous models; and they bring with them extensive theoretical underpinnings.
3. Graph models are easily implemented. In fact, they are special cases of logic models and are naturally implemented via logic programming.
4. Graph models are familiar to many people and afford excellent visualizations.

Procedures. A procedural or computational model seeks to account for—to explain or illustrate—its target system [Kim03]. Standard, equational models have the same purpose. For that matter, so do logic models and graph models. We identify a system of interest; we develop a model or formal representation (including the proposed correspondence between the model and the target system); and then we observe the behavior of the two. If the model's behavior matches the target system's behavior in the right ways, we congratulate ourselves on a good job and are prepared to rely upon the model in making predictions and in otherwise directing our own behavior.

Perhaps the clearest example of an explicitly *procedural* explanation is the Darwinian theory of evolution. There are no equations in *The Origin of Species* and the theory has not been successfully axiomatized. Equational models are used—extensively—to model the behavior of evolutionary systems, but not to model the basic theory itself. See for details the quoted passage on the next page, taken from [Kim03].

It is by now standard to note that evolutionary theories of adaptation (e.g., Lamarck's and that of Charles's grandfather Erasmus Darwin) predated *The Origin of Species* (1859). Darwin's achievement was not to originate a theory of evolution. Rather it was (with Alfred Russell Wallace) to propose a workable (and largely correct) mechanism or *procedure* by which evolution comes about. In fact, Darwin did not use the term evolution in the first edition of the *Origin.* Instead, he consistently wrote of his theory of "descent with modification by natural selection." Most fundamentally (and sufficient for present purposes) Darwin (and Wallace) put together three ideas or observations:

1. *Profusion* of individuals with *variance* of traits
   Every species has the capability to, and tends to, produce more offspring each generation than can possibly survive and reproduce. The individuals so produced are not all identical.
2. *Selection* among the variants
   Natural selection operates on populations of varying individuals, selecting for properties favorable to survival and reproduction.
   The individuals in a species vary in many different ways, including their capacity, at least in expectation, for survival and reproduction.
3. *Reproduction* of variants favored by selection, with *inheritance* of favorable traits.
   Inheritance may be approximate and far from perfect. What is required is a heritable association between the traits favorable to the parents and the traits of their offspring.

Given such a regime—of profusion with variance of traits, selection by traits, and reproduction with inheritance—it is nearly inevitable that evolution or "descent with modification by natural selection" will occur. Darwin and Wallace claimed that in fact it did routinely and that this process has in the main been responsible for adaptation and speciation. Such, boiled down for present purposes, is the (biological) theory of evolution by natural selection.[a] So successful has this theory become that now when we say something is an evolutionary theory or account, we *mean* it appeals to, or posits, a profusion-selection-reproduction process.

[a] For a more detailed analysis see [Kim80].

3 Themes

A number of themes, or general problems, have been prevalent among various FMEC contributors. It is convenient to list them as follows:

- Representation
- Inference
- Learning

Representation. The first and most pervasive theme was the impetus for Ronald M. Lee's seminal Ph.D. dissertation, describing his CANDID system.[3] Lee's problem was how to represent business entities, including actors, economic objects, financial entities and instruments, and business documents (such as invoices and purchase orders). We might call this the *representation theme in e-commerce*. It leads to conceptual analysis with formal presentation of the results.

Lee's thought was to model the gamut of business entities using a formal, broadly logical language. Lee's work is extensive but incomplete and often programmatic. His reliance on Montague semantics may pose severe problems for implementation. Nevertheless, his results and insights are important. Further, the inherent subtlety and difficulty, along with the potential usefulness, of the project may fairly be said to have inspired a body of research that continues to this day and that has been central to the FMEC community. Much has been achieved and much remains to be done. Lee's dissertation has until now appeared only in working paper form. We are very pleased to present an updated version in this volume. The new CANDID manuscript constitutes three chapters in Part I, pages 101–176.

Inference. Inference is the second theme that has occupied the attention of FMEC researchers. There has been a shared vision, not often explicitly articulated in print,[4] of having general, widely-used deductive databases for reasoning in electronic commerce about obligations, time, actions, contracts, trade procedures, and so on, and to be able to do so defeasibly.

Learning. Interest in learning, the third major theme in FMEC research, has been occasioned by the shared vision of having very powerful—AAAA ("anything, any time, anywhere, anywise")—artificial agents present on the Internet, doing business on our behalf. Because these agents will be interacting with other agents (artificial or not), they will be acting in strategic or game-theoretic contexts. Because classical game theory has little to say about how to pick a strategy for play in a game and because many of the

[3] See [Lee80].
[4] But see [KM93a].

strategic contexts of electronic commerce will involve repetitions of simpler games, having the artificial agents learn which strategies to play becomes imperative. Understanding what agents will learn under various regimes of play and kinds of learning is very much an open issue, although significant results are reported in this volume.

4 Topics

Quite a number of more specific topics—falling within the general themes—have been investigated by the FMEC community. We now provide, with brief discussion, a representative list.

- Electronic data interchange.
  The term *electronic data interchange* is used as an abstract, more general term for EDI (Electronic Data Interchange) in the sense denoting use of specific protocols, such as X12 and UN/EDIFACT, or protocols like them. The topic of electronic data interchange covers the general problem of designing effective and powerful protocols (or more generally languages) for conduct of business. Not only must standard documents, such as invoices, purchase orders, and bills of lading, be represented, but conversations must be held, trading agreements made and so on. Much of the FMEC work on this topic is self-described as working on an FLBC, a Formal Language for Business Communication.
- Electronic contracting.
  This is closely related to the electronic data interchange topic. How can artificial agents effectively compose, evaluate, monitor, and be directed by contracts in electronic form? Much of the FMEC work has focused on analysis and representation of trade procedures, with the view that libraries might be created upon which agents might draw in negotiating deals. An important part of the electronic contracting topic is the *first trade problem*. A main cause of the high expense of EDI is the cost of setting up the contractual arrangements for making the first trade. Once that is done, if done well, the incremental costs of EDI (more generally, electronic data interchange) may be quite modest.
- Speech acts.
  Contracts and other business documents (the usual: invoices, purchase orders, bills of lading, etc.) are propositional. They make assertions, issue promises, give commands, and so on. Individual assertions, promises, commands et cetera are widely recognized as *speech acts*. They *say* things and they *do* things. Speech acts are essential to business communication, yet their logical structure is problematic. FMEC researchers have been much concerned with exploring how speech acts should be represented formally, so that artificial agents may undertake them and reason about them.

- Special logics.
  FMEC researchers have investigated logics of defeasible reasoning, action, deontic reasoning, modal reasoning, epistemic reasoning, temporal reasoning, institutional power, and reasoning about speech acts—all as applicable to electronic commerce.
- System and process modelling.
  FMEC researchers have expended much attention on developing graph models (usually) or logic models (less usually) for representing workflows, trade procedures, and other processes of import to electronic commerce, as well as systems in which such processes occur.
- Strategy formation.
  FMEC researchers have produced a now extensive body of work (the most recent of which is contained in this volume) exploring how various computational learning regimes fare in finding effective strategies for play in repeated games.
- Computational discovery.
  In the strategy formation work, FMEC researchers have investigated how computations may discover effective strategies. More generally, FMEC researchers have explored related learning methods for computational discovery in such areas as data mining[5] and investment policies.[6]

5 A Brief Guide to the Volume

With these remarks serving as a map, we may make short work of characterizing the twenty papers of this volume. The result will guide the reader. The papers are divided into four parts of the volume.

Part I. "Representation: Objects, Processes & Policies" contains seven papers:

- Alan Abrahams, David Eyers, and Jean Bacon, "Practical Contract Storage, Checking, and Enforcement for Business Process Automation," pages 33–77.

This paper describes an extensive prototype implementation, in Java and a relational database, of a logic-based language for conducting commerce (Kimbrough's FLBC). In terms of our FMEC framework, the paper focuses on a procedural implementation of a logic formalism. It addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topics of electronic data interchange, electronic contracting, speech acts, and special logics.

[5] E.g., [PT02].
[6] E.g., [Wu97].

- Aldo de Moor and Hans Weigand, "Legitimacy Checking in Communicative Workflow Design," pages 79–99.

This paper introduces what the authors call an *extended workflow loop*. Using this idea as the basic unit of analysis, they introduce the concept of workflow loop norms, which are grounded in internal control theory. The paper develop this and related ideas, and demonstrates their use with examples. In terms of our FMEC framework, the paper focuses on graphs for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topic of system and process modelling.

We have already discussed the three chapters on the CANDID system:

- Ronald M. Lee, "CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part I: Syntax & Formal Semantics of CANDID," pages 101–143.
- Ronald M. Lee, "CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part II: Formal Description of Economics Actors and Objects," pages 145–158.
- Ronald M. Lee, "CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part III: CANDID Specification of Financial Concepts," pages 159–176.

We warn the reader that these chapters are dense with notation. The underlying ideas, however, are significant and merit the effort to take them in.

- Ronald M. Lee, "Performatives, Performatives Everywhere but Not a Drop of Ink," pages 177–200.

This fourth paper by Lee, like the previous three and like the next paper by Kimbrough, is from the FMEC archives. Written in the mid-1990s, it has never been published, although it has circulated widely and influenced FMEC work.

The paper observes that the feasibility of open, flexible electronic commerce relies heavily on the effective management of documentary procedures, i.e., the sequence by which (structured) business documents are exchanged among contracting parties. These communications are *performative* (versus informative) in that the act of communicating itself is a social action that alters the contractual, legal, or ownership relationship among the parties. The paper goes on to discuss the problems of supporting performative communications and how they might be handled in principle. In terms of our FMEC framework, the paper focuses on logic for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topic of speech acts (speech act theory was developed in part as a response to the recognition that utterances may be performative), electronic data interchange, and electronic contracting.

- Steven O. Kimbrough, "EDI, XML, and the Transparency Problem in Electronic Commerce," pages 201–227.

This paper, written in the mid-1990s and presented at a conference, attempts to put to rest a confusion still present in some quarters. Standard EDI protocols (including the X12 and UN/EDIFACT series) have repeatedly been criticized for poor design, incoherent or absent semantics, and much else. XML has been touted by some as a proper remedy. The paper argues that this is a confusion. While there are many positive aspects of XML, the claim that its tagging system allows documents to be semantically "self-describing" is misleading and overblown. XML may well be useful as a part of the solution to the problem of making documents and messages semantically transparent, but its role is at best marginal for this problem. The paper goes on to provide an account of what a proper solution would look like. In terms of our FMEC framework, the paper focuses on logic for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topics of speech acts, electronic data interchange, and electronic contracting.

Part II. "Applications" has two papers:

- Vera Kartseva and Yao-Hua Tan, "Designing Control Mechanisms for Value Exchanges in Network Organisations," pages 231–246.

This paper proposes a model for monitoring contract compliance, based in part on concepts from deontic logic. The result is a design tool for modelling violations of obligations, which can be used in contract drafting and contingency planning for inter-organisational collaboration in network organisations. In terms of our FMEC framework, the paper focuses on logic and graphs (of organizational procedures) for its formalism. The paper addresses the central topic of strategy formation, has as a key theme representation in electronic commerce, and offers much on the topic of system and process modelling.

- Max Boisot, Ian MacMillan, Kyeong Seok Han, Casey Tan, and Si Hyung Eun, "Sim-I-Space: An Agent-Based Modelling Approach," pages 247–294.

This paper describes Sim-I-Space, an agent-based model that operationalises key features of a conceptual framework: the Information-Space (I-Space). The I-Space relates the speed and extent of information flows between agents to how far their messages have been structured through acts of codification and abstraction. The more structured a message, the faster and more extensively it diffuses to other agents—intentionally or not. This concept is used to discover and analyze strategic options for businesses, contingent upon the

diffusion of knowledge. In terms of our FMEC framework, the paper focuses on procedures for its formalism. The paper addresses the central topic of strategy formation, and has as a key theme representation in electronic commerce.

Part III. "Communication" contains three papers:

- Steven O. Kimbrough and Yinghui (Catherine) Yang, "On Representing Special Languages with FLBC: Message Markers and Reference Fixing in SeaSpeak," pages 297–324.

SeaSpeak is "English for maritime communications." It is a restricted, specially-designed dialect of English used in merchant shipping and accepted as an international standard. This paper discusses, in the context of SeaSpeak, two key problems in the formalization of any such restricted, specially-designed language, viz., representing the illocutionary force structure of the messages, and formalization of such reference-fixing devices from ordinary language as pointing and use of demonstratives. In terms of our FMEC framework, the paper focuses on logic for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topic of speech acts.

- Andrew J.I. Jones and Steven O. Kimbrough, "A Note on Modelling Speech Acts as Signalling Conventions," pages 325–342.

This paper presents a fully formal integration of Jones's logical theory of speech acts as signalling conventions with Kimbrough's Formal Language for Business Communication (FLBC). The paper distinguishes between 'intentionist' accounts of speech acts and 'conventionist' accounts. In contradistinction to essentially all work in agent communication languages, Jones's theory of speech is thoroughly conventionist. The paper argues that this is a strong advantage for the theory and demonstrates that Kimbrough's FLBC fits comfortably and naturally with Jones's theory of speech acts. In terms of our FMEC framework, the paper focuses on logic for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topics of speech acts, special logics, and electronic data interchange.

- Scott A. Moore, "Dynamic Conversation Structures: An Extended Example," pages 343–360.

The subject of this paper is the important one of modelling and design of conversations between communicating agents. The paper provides a detailed look at a moderately complex conversation as represented by a finite state machine, a representation used by an established agent communication system. Various representational methods are compared and discussed. The paper

concludes with a demonstration of how a multi-agent conversation policy can be used to control the flow of messages, contrasts this with how messages are handled via an inference-based process, and shows how the inference-based processing can be integrated with the policy-based handling in order to deal with exceptions to the policy. In terms of our FMEC framework, the paper focuses on graphs for its formalism. The paper addresses the principal themes of representation and inference in electronic commerce, and it has much to say on the topic of electronic data interchange.

Part IV. "Agents and Strategic Interactions" has eight papers:

- Scott A. Moore and Kurt Demaagd, "Investigating the Value of Information and Computational Capabilities by Applying Genetic Programming to Supply Chain Management," pages 363–391.

This paper presents the design of an innovative system for simulating agents in supply chains. The agents undertake computational search for effective strategies using genetic programming. In terms of our FMEC framework, the paper focuses on procedures for its formalism. The paper addresses the principal theme of learning in electronic commerce, and it has much to say on the topics of electronic data interchange, strategy formation, and computational discovery.

- Olga Streltchenko, Yelena Yesha, and Timothy Finin, "Multi-Agent Simulation of Financial Markets," pages 393–419.

This paper discusses the principal reasons for, and prospective opportunities of, simulating financial markets using an architecture based on artificial agents. The paper then discusses in detail the design and architecture of a simulator for financial markets. In terms of our FMEC framework, the paper focuses on procedures (principally, reinforcement learning) for its formalism. The paper addresses the principal theme of representation in electronic commerce, and it has much to say on the topic of strategy formation.

- Alex K. Chavez, "Adaptive Agents in Coalition Formation Games," pages 421–443.

Coalition formation games form an important subclass of mixed-motive strategic situations, in which players must negotiate competitively to secure contracts. This paper compares the performance of two learning mechanisms, reinforcement learning and counterfactual reasoning, for modeling play in coalition formation games. In terms of our FMEC framework, the paper focuses on procedures (principally, reinforcement learning) for its formalism. The paper addresses the principal theme of representation in electronic commerce, and it has much to say on the topic of strategy formation.

- Jim R. Oliver, "On Learning Negotiation Strategies by Artificial Adaptive Agents in Environments of Incomplete Information," pages 445–461.

This paper examines automated negotiation by artificial adaptive agents, which holds great promise for electronic commerce. Difficult practical issues remain unresolved, however. Published studies of learning of negotiation strategies by agents have been based on artificial environments that include complete payoff information for both sides of the bargaining table. This is not realistic in applied contexts. This paper considers the case of a seller who knows its own preferences over negotiation outcomes but who has only limited information about the private values of each customer. In terms of our FMEC framework, the paper focuses on procedures (principally, genetic algorithms) for its formalism. The paper addresses the central topic of strategy formation, particularly in the context of negotiation.

- Steven O. Kimbrough, Ming Lu, and Ann Kuo, "A Note on Strategic Learning in Policy Space," pages 463–475.

This paper introduces learning in policy space, in distinction to the usual learning in state space, for agents in games. Instead of, as in most studies, the agents learning to associate plays with the recent history of play, agents here learn which policies to play for a period of time, based on how well they perform compared to other policies. The paper examines play in a number of repeated 2×2 games and finds that policy-space learning agents are generally more effective than state-space learning agents in extracting wealth from the game. In terms of our FMEC framework, the paper focuses on procedures (reinforcement learning in several senses) for its formalism. The paper addresses the central topic of strategy formation.

- Steven O. Kimbrough, Ming Lu, and Frederic Murphy, "Learning and Tacit Collusion by Artificial Agents in Cournot Duopoly Games," pages 477–492.

This paper explores learning by artificial agents in repeated play of Cournot duopoly games. The agents' employ policy-space learning regimes. The resulting behavior is markedly different from behavior predicted by classical economics for the single-shot (unrepeated) Cournot duopoly game. In repeated play under this learning regime, agents are able to arrive at a tacit form of collusion and set production levels near to those for a monopolist. The paper notes that Cournot duopoly games are reasonable approximations for many real-world arrangements, including hourly spot markets for electricity. In terms of our FMEC framework, the paper focuses on procedures (reinforcement learning in several senses) for its formalism. The paper addresses the central topic of strategy formation.

- Fang Zhong, "A Note on Working Memory in Agent Learning," pages 493–507.

The existing literature reports mixed findings on the effects of the amount of working memory on the effectiveness of agents in strategic contexts. In this note, Zhong describes an intelligent agent system in which three agents, one buyer and two bidders, play an Exchange game repeatedly. The buyer agent decides whether to list a request for proposal, while the bidders bid for it independently. The paper finds that the relationship between working memory and the effectiveness of the agents has an inverted U shape, i.e., there seems to be an optimal memory size. When agents with different memory sizes are mixed together, agents with the same amount of working memory generate the most efficient outcome in terms of total payoffs. In terms of our FMEC framework, the paper focuses on procedures (reinforcement learning) for its formalism. The paper addresses the central topic of strategy formation.

- Sofia Chajadine, Daniel Mack, and Aaron Jeffrey Slan, "Investigations of Granularity and Payoffs in 2×2 Games under Replicator Dynamics," pages 509–527.

This paper describes an investigation of several 2×2 games in iterated form. Players play the games repeatedly and are limited to mixed strategies, with particular actions chosen probabilistically. The games investigated include Prisoner's Dilemma, Chicken, and Stag Hunt in various forms. The reward structure and the granularity of the games—number of games played per generation in the replicator dynamics—are the main factors investigated, with results that contravene existing studies that neglect these factors. In terms of our FMEC framework, the paper focuses on procedures (replicator dynamics) for its formalism. The paper addresses the central topic of strategy formation.

6 Upwards and Onwards

This completes our survey of FMEC research prior to and including this volume. What does or should the future hold? In what directions is the community likely to go?

Engaging more directly with practice has been a much-discussed goal of the group. One considered view is that "next generation enterprise computing" and agent-mediated electronic commerce, two important concepts under wider development, could benefit from achievements of FMEC research. Both concepts are by nature incremental (although they are not advertised that way!). This facilitates a trial-and-feedback arrangement between ideas originated in the FMEC community and the greater world of practice and use (aka: the 'real' world). Proxy bidding, in which artificial agents participate in

auctions while closely supervised by their human masters, is already in use and may well be a good model for deployment of other ideas.

Stepping back and taking perspective, the themes and topics addressed by the FMEC community are important, are interesting, and afford nearly unlimited scope for further investigation and development. Moreover, the ('non-standard') modelling formalisms in use—logic, graphs, and procedures—have been productive and show no signs of exhaustion. Clearly, 'broader and deeper' is a fine option. There is ample reason to continue on the path that got us here. That said, there is every reason to think that new ideas and new directions will also be found and welcomed. What might they be? We do not presume to say. Instead, we close this essay with a speculative idea, which has been discussed at times within the FMEC community. We conclude, then, in the spirit of asking a question—Is this a good idea?—rather than of predicting, let alone mandating, new developments.

Embodied Research

By design, open source software development resembles scientific research in certain ways. Both are open in the sense of being public and in the sense of inviting participation by anyone having a genuine contribution to make. Both are conducted by loosely organized, modestly hierarchical communities of volunteers. Both produce public goods, which may be exploited by anyone, including non-contributors. Both command general approbation in virtue of the open, public, and objective processes that underlie them, including an open market for testing and validation. Both motivate and reward their participants largely by recognition, by the satisfaction due to making an impact, by access to information and other contributors "on the leading edge", and by whatever immediate value the partcipants get from the resulting work products.

Scientific research and open source development differ crucially in that one is *research* and the other is *development*. Roughly, research aims at producing knowledge or know-how. Development aims at producing products that are useful and used, products in which knowledge is embodied (to borrow a term from patenting). Again roughly, the work product of research is symbolic and representational—axioms, models, equations, texts, theorems, algorithms, and so forth. The work product of development is tangible and embodied, locatable in space as well as time. Software is of course a borderline case. Even so, we may think of 'research software' as embodying ideas and demonstrating their feasibility. Production or 'commercial grade' software is more unproblematically an embodiment of its originating, abstract ideas (whether or not produced by research).

Open source development projects are normally *not* driven by scientific research. They are ascientific. Their usual purpose is to produce a useful product of some sort (Web server, text editor, browser, etc.), unmotivated by any scientific or research-oriented considerations. The question we wish to raise is:

- Under what conditions would it be attractive and sensible to undertake open-source-like development (of software or of other forms of content) for the sake of furthering scientific (and possibly purely practical) ends?

We shall call such a venture an *embodied research* programme; it aims at producing both scientific advances *and* useful, production-grade products, such as software and courseware. The concept itself raises any number of new questions. We will briefly address a few of them here.

- What sorts of research projects or topics might be suitable for embodied research?

Properties of promising embodied research projects involving software (or courseware) development would seem to include the following:

1. The envisioned software is not otherwise available or likely to be supplied by the market.
2. The envisioned software is sufficiently simple that it can be created with academic-level resources and capabilities. This suggests 'lightweight' applications for lower-end users.
3. The envisioned software is demonstrably valuable to a significant client community, which is unable to afford commercial purchase of the capability but which is ready and able to participate in the research and development efforts.
4. The envisioned software can be produced in individually useful increments.
5. The envisioned software is useful for testing interesting research concepts, or for answering research questions.
6. The envisioned software is useful for demonstrating interesting research concepts and results, and for communicating ideas to interested parties, including funding organizations and potential clients.
7. The associated research issues are not exhaustible in the near term. The research topic must afford depth, challenge, and longevity to the research programme. It must also be inherently interesting from a research perspective.
8. The embodied research project must afford partial, limited contributions by the participants. Specialized contributions should be welcome and legitimately appropriate; the project is decomposable, both in research and in development.

- Are there any plausible examples of, or candidates for, such topics?

Plausible examples include:

1. FMEC: formal modeling for electronic commerce.
   Embodied agent communication languages, e.g., via ebXML. This presents an apt challenge for theoretical analysis, and for connection to automated (presumably logical) reasoning. Software affords a demonstration testbed and could be used by SMEs (small and medium-sized entities) and firms in developing countries.
2. Information retrieval, mining, and management for communities of museums, hospitals, local governments, and other non-profit organizations.
3. Optimization software for researchers and practitioners (and for educators).
   An excellent start has been achieved by COIN (www.coin-or.org), which is perhaps the closest existing embodiment to the embodied research concept we are attempting to vet here.
4. Agent-based software for research and teaching purposes.
   An intriguing, although nascent, example is www.agentbasedis.org (Association for Information Systems, Special Interest Group on Agent-Based

Information Systems). Note as well the Swarm Development Group at www.swarm.org.
VATSIM (www.vatsim.net) offers another, also successful, model. From their "About" page:

> The Virtual Air Traffic Simulation Network, known as VATSIM.net or "VATSIM" was created in 2001 by a group of individuals who came together with a goal of creating an organization which truly served the needs of the flight simulation and online air traffic control community. With an eye towards more than just providing a network of computers for users to log into, VATSIM is an online community where people can learn and, at the same time, enjoy the pastimes of flight simulation and air traffic control simulation all while making new friends from all over the world.

There now exists a vibrant community organized around VATSIM, as well as remarkably complex and robust software supporting these activities.

- What incentives would or could scientific (normally academic) researchers have to contribute to an embodied research program?

Possibilities:

1. Recognition, impact
2. Access to leading ideas and people; stimulus for research ideas
3. Prospect for joint proposals for research funding
4. Expectation of being able to use the developed software for teaching, research, or consulting
5. Special interest in furthering the subject matter, e.g., e-business for SMEs (small and medium-sized entities) and developing countries
6. Providing students with opportunities to contribute in ways that will further their careers.
7. Interest in the open source process itself
8. As a vehicle for furthering and embodying existing ideas and expertise
9. Exposure, publicity, networking
10. Useful feedback and testing of ideas

- How would one organize and initiate such a venture?

This is good question. It should be answered in more detail once the plausibility case is established. But roughly: (1) organize on the model of a journal: editorial board of leading contributors, find contributing clients; (2) carefully initialize to guarantee some immediate successes (e.g., produce a body of work and make it the initial contribution to the project); (3) systematically think through mechanisms for rewarding and incenting participation.

Regarding point (3), at least in some computer science departments software contributions can be 'counted as' valid forms of publication. Perhaps in time this convention could be strengthened and extended in scope. Also, the leadership of an embodied research project (e.g., the editorial board) could do much to arrange for opportunities for refereed publication.

- Could you be a bit more specific on what the next steps are?

  Yes, but this is best left open to a general discussion. May it continue.

7 Acknowledgements

We wish to thank Ronald M. Lee, Robin Lougee-Heimer, Scott A. Moore, Marek Sergot, and Paul Weinberg for productive conversations contributing to and exploring the ideas here recorded. Of course, we insist on complete credit for everything here that is in error, misguided, or simply stupid.

A FMEC Bibliographic History

The birth of the FMEC community may be dated to the Logic Modelling minitrack at the 20th Hawaii International Conference on System Sciences (HICSS), held in January 1987, and organized that year by Ronald M. Lee and Steven O. Kimbrough. At HICSS, the annual Logic Modelling sessions, and later the FMEC minitrack itself, were supported throughout by Ralph H. Spague, Jr. We are pleased to again acknowledge his contributions, for which we are most sincerely grateful.

Special, independent FMEC workshops have been held at:

- EURIDIS (Erasmus University Research Institute for Decision and Information Systems), Erasmus University, Rotterdam, The Netherlands, in 1999
- Drexel Univeristy and the University of Pennsylvania, Philadelphia, USA, in 2000
- The University of Oslo, Oslo, Norway, in 2001
- Łodz, Poland, in 2002

Papers from the FMEC community may be found in the HICSS *Proceedings*, beginning in 1987. The Logic Modelling and FMEC minitracks have concentrations of the papers, but other minitracks hold them as well. To date, 21 Ph.D. theses have been produced in the FMEC community. They are listed in Appendix A.1, below.[7] The FMEC community has so far published 10 special issues of journals, comprising 59 refereed articles. These are listed in Appendix A.2, below. We are most grateful for the support of the editors of the host journals:

- Melvin F. Shakun, *Group Decision and Negotiation*
- Andrew Whinston, *Decision Support Systems*
- Vladimir Zwass, *Journal of Management Information Systems* and *International Journal of Electronic Commerce*

[7] We apologize if we have missed any.

The open literature contains a large number of papers arising out of the FMEC community. The comprehensive References section of this volume is a good source for locating these papers. The home pages and *curricula vitae* of the FMEC authors may be consulted for additional papers.

Besides HICSS, members of the FMEC community have been, and continue to be, active in a number of conferences and workshops, notably: ICAIL (International Conference on Artificial Intelligence and Law), DEON (International Workshop on Deontic Logic in Computer Science), the Bled Electronic Commerce Conference, and LAP (Language-Action Perspective on Communication Modelling, International Working Conference).

A.1 FMEC Ph.D. Theses

1. *CANDID: A Logical Calculus for Describing Financial Contracts* by Ronald M. Lee, University of Pennsylvania, USA, 1980, [Lee80].
2. *A Logic Model for Model Management: An Embedded Languages Approach* by Hemant K. Bhargava, University of Pennsylvania, USA, 1990, [Bha90].
3. *Timed Coloured Petri Nets and their Application to Logistics* by W.M.P. van der Aalst, Eindhoven Technical University, The Netherlands, 1992, [Aal92].
4. *Schematic Evaluation of Internal Accounting Control Systems* by Kuo-Tay Chen, University of Texas at Austin, USA, 1992, [Che92].
5. *Contracting on a Performative Network: Using Information Technology as a Legal Intermediary* by Sandra D. Dewitz, University of Texas at Austin, USA, 1992, [Dew92].
6. *A Formal Model for Maintaining Consistency of Evolving Bureaucratic Policies: A Logical and Abductive Approach* by Kay Liang Ong, University of Texas at Austin, USA, 1992, [Ong92]
7. *Probabilistic and Defeasible Reasoning Using Extended Path Analysis* by Stephen F. Roehrig, University of Pennsylvania, USA, 1992, [Roe92].
8. *A Formal Representation of Normative Systems: A Defeasible Deontic Reasoning Approach* by Young Ryu, University of Texas at Austin, USA, 1992, [Ryu92].
9. *Saying and Doing: Uses of Formal Languages in the Conduct of Business* by Scott A. Moore, University of Pennsylvania, USA, 1993, [Moo93].
10. *Theory and Applications of Argumentation Support Systems* by Hua Hua, University of Pennsylvania, USA, 1995, [Hua95].
11. *Formal Theories of Rights* by Henning Herrestad, University of Oslo, Norway, 1996, [Her96].
12. *On Artificial Agents for Negotiation in Electronic Commerce* by Jim R. Oliver, University of Pennsylvania, USA, 1996, [Oli96b].
13. *The Structure of Business Communication: Theory, Model and Application* by Victor Emil van Reijswoud, Delft University, The Netherlands, 1996, [vR96].

14. *Designing Trustworthy Trade Procedures for Open Electronic Commerce* by Roger Bons, Erasmus University, The Netherlands, 1997, [Bon97].
15. *Normative Structures in Natural and Artificial Systems* by Cristen Krogh, University of Oslo, Norway, 1997, [Kro97].
16. *Managing Complex, Open, Web-Deployable Trade Objects* by Hung Wing, University of Queensland, Australia, 1997, [Win97].
17. *Using Genetic Algorithms to Determine Near-Optimal Pricing, Investment and Operating Strategies in the Electric Power Industry* by Dongjun (D.J.) Wu, University of Pennsylvania, USA, 1997, [Wu97]
18. *Logic-Based Tools for the Analysis and Representation of Legal Contracts* by Aspassia Daskalopulu, Imperial College, University of London, UK, 1999, [Das99].
19. *Information Sharing among Ideal Agents* by Alessio R. Lomuscio, School of Computer Science, University of Birmingham, UK, 1999, [Lom99].
20. *Developing and Executing Electronic Commerce Applications with Occurrences* by Alan S. Abrahams, University of Cambridge, UK, 2002, [Abr02].
21. *Strategy as Valuation* by Christina Fang, University of Pennsylvania, Philadelphia, PA, USA, [Fan03].

A.2 FMEC Special Issues Bibliography

March 1988 *Decision Support Systems*, volume 4, number 1, 1988.

1. "Logic Modeling: A Tool for Management Science" by Steven O. Kimbrough and Ronald M. Lee [KL88].
2. "Requirements Analysis Assisted by Logic modelling" by Peter C. Scott [Sco88].
3. "A Logic Model for Electronic Contracting" by Ronald M. Lee [Lee88b].
4. "A Quantitative Approach to Logical Inference" by J.N. Hooker [Hoo88].
5. "Spatial Imbedding for Linear and for Logic Structures" by Robert G. Jeroslow [Jer88].
6. "Logic modelling with Partially Ordered Preferences" by George R. Widmeyer [Wid88].
7. "Defeasible Reasoning and Decision Support Systems" by Donald Nute [Nut88].
8. "Why Nonmonotonic Logic?" by Steven O. Kimbrough and Fred Adams [KA88].
9. "A Conditional Logic for Defeasible Beliefs" by Marvin Belzer and Barry Loewer [BL88].

May 1990 *Decision Support Systems*, volume 6, number 2, 1990.

10. "The Sensitivity Properties of Hierarchical Logic-Based Models" by Robert W. Blanning [Bla90].
11. "On Representation Schemes for Promising Electronically" by Steven O. Kimbrough [Kim90].
12. "A Logic modelling Language for Automated Model Construction" by Ramayya Krishnan [Kri90].
13. "Controlling Expert System Recommendations with Defeasible Logic" by Donald Nute, Robert I. Mann, and Betty F. Brewer [NMB90].
14. "Meta-Interpreters for Rule-Based Inference under Uncertainty" by Shimon Schocken and Tim Finin [SF90].
15. "Reasoning with Preferences and Values" by George R. Widmeyer [Wid90].

February 1994 *Decision Support Systems*, volume 11, number 2, 1994.

16. "EVID: A System for Interactive Defeasible Reasoning" by Robert L. Causey [Cau94].
17. "Defeasible Reasoning in Law" by Sandra D. Dewitz, Young Ryu, and Ronald M. Lee [DRL94].
18. "Ordered Logic: Defeasible Reasoning for Multiple Agents" by P. Geerts, D. Vermeir, and D. Nute [GVN94].
19. "Bayesian Logic" by K. A. Andersen and J. N. Hooker [AH94].
20. "A Relational Algebra for Propositional Logic" by Robert W. Blanning [Bla94].
21. "Text Editing and Beyond: A Study in Logic modelling" by Michael Bieber and Thomás Isakowitz [BI94].

Summer 1997 *International Journal of Electronic Commerce*, volume 1, number 4.

22. "Formal Aspects of Electronic Commerce: Research Issues and Challenges" by Steven O. Kimbrough and Ronald M. Lee [KL97].
23. "On Designing a Language for Electronic Commerce" by Michael A. Covington [Cov97].
24. "Artificial Agents Learn Policies for Multi-Issue Negotiation" by Jim R. Oliver [Oli97a].
25. "Designing a Market for Quantitative Knowledge" by Georg Geyer, Christoph Kuhn, and Beat Schmid [GKS97].
26. "Electronic Commerce in Decision Technologies: A Business Cycle Analysis" by Hemant K. Bhargava, Ramayya Krishnan, and Rudolf Müller [BKM97].
27. "The Development of FEDI in Switzerland: A Life-Cycle Approach" by Ivo Cathomen and Stefan Klein [CK97].

March 1998 *Decision Support Systems*, volume 22, number 3, 1998.

28. "Speech Acts, Electronic Commerce, and KQML" by Michael A. Covington [Cov98].
29. "Categorizing Automated Messages" by Scott A. Moore [Moo98].
30. "A Logic-Based modelling of Resource Consumption and Production" by Young U. Ryu [Ryu98].
31. "On Hypermedia-Based Argumentation Decision Support Systems" by Gary H. Hua and Steven O. Kimbrough [HK98].
32. "Defeasible Logic Graphs: I. Theory" by Donald Nute and Katrin Erk [NE98].
33. "Defeasible Logic Graphs: II. Implementation" by Donald Nute, Zachary Hunter, and Christopher Henderson [NHH98].

Winter 1998–99 *International Journal of Electronic Commerce*, volume 3, number 2.

34. "Framework for Specifying, Building, and Operating Electronic Markets" by Markus A. Lindemann and Beat F. Schmid [LS99].
35. "Formal Language for Business Communication: Sketch of a Basic Theory" by Steven O. Kimbrough [Kim99].
36. "Meta-Patterns for Electronic Commerce Transactions Based on the Formal Langauge for Business Communication (FLBC)" by Hans Weigand and Willem-Jan van den Heuvel [Wv99].
37. "Specifying Deadlines with Continuous Time Using Deontic and Temporal Logic" by Frank Dignum and Ruurd Kuiper [DK99].
38. "A Logical Model of Directed Obligations and Permissions to Support Electronic Contracting" by Yao-Hua Tan and Walter Thoen [TT99].
39. "Distributed Electronic Trade Scenarios: Representation, Design, Prototyping" by Ronald M. Lee [Lee99].

Fall 2000 *International Journal of Electronic Commerce*, vol. 5, no. 1.

40. "Artificial Agents for Discovering Business Strategies for Network Industries" by D.J. Wu [Wu00].
41. "Pricing and Product Design: Intermediary Strategies in an Electronic Market" by Hemant K. Bhargava, Vidyanand Choudhary and Ramayya Krishnan [BCK00].
42. "A Formal Analysis of Auditing Principles for Electronic Trade Procedures" by Roger W.H. Bons, Frank Dignum, Ronald M. Lee, and Yao-Hua Tan [BDLT00].
43. "On Lean Messaging with Unfolding and Unwrapping for Electronic Commerce" by Steven O. Kimbrough and Yao-Hua Tan [KT00].

44. "KQML and FLBC: Contrasting Agent Communication Languages" by Scott A. Moore [Moo00a].

July 2002 *Decision Support Systems*, volume 33, number 3, 2002.

45. "On the Concept of Trust" by Andrew J.I. Jones [Jon02].
46. "Formal Aspects of a Generic Model of Trust for Electronic Commerce" by Yao-Hua Tan and Walter Thoen [TT02].
47. "Cross-Organizational Workflow Integration Using Contracts" by Hans Weigand and Willem-Jan van den Heuvel [Wv02].
48. "Computational Aspects of the FLBC Framework" by Aspassia Daskalopulu and Marek Sergot [DS02].
49. "Automated Generation of Electronic Procedures: Procedure Constraint Grammars" by Ronald M. Lee [Lee02].
50. "Knowledge Refinement Based on the Discovery of Unexpected Patterns in Data Mining" by Balaji Padmanabhan and Alexander Tuzhilin [PT02].
51. "Computers Play the Beer Game: Can Artificial Agents Manage Supply Chains?" by Steven O. Kimbrough, D.J. Wu, and Fang Zhong [KWZ02].
52. "Cooperation in Multi-Agent Bidding" by D.J. Wu and Yanjun Sun [WS02].

November 2002 *Group Decision and Negotiation*, vol. 11, no. 6, 2002.

53. "Cooperative Agent Systems: Artificial Agents Play the Ultimatum Game" by F. Zhong, Steven O. Kimbrough, and D.J. Wu [ZKW02].
54. "On Adaptive Emergence of Trust Behavior in the Game of Stag Hunt" by Christina Fang, Steven O. Kimbrough, Stefano Pace, Annapurna Valluri, and Zhiqiang Zheng [FKP+02].
55. "Evidence-Based Electronic Contract Performance Monitoring" by Aspassia Daskalopulu, Theo Dimitrakos. and Tom Maibaum [DDM02].
56. "A Software Implementation of Kimbrough's Disquotation Theory for Representing and Enforcing Electronic Commerce Contracts" by Alan S. Abrahams and Jean M. Bacon [AB02b].

January 2003 *Group Decision and Negotiation*, vol. 12, no. 1, 2003.

57. "B2B Negotiation Support: The Need for a Communication Perspective" by Hans Weigand, Mareike Schoop, Aldo de Moor, and Frank Dignum [WSdMD03].
58. "A Classification Scheme for Negotiation in Electronic Commerce" by Alessio R. Lomuscio, Michael Wooldridge, and Nicholas R. Jennings [LWJ03].

59. "The Effects of Personal and Group Level Factors on the Outcomes of Simulated Auditor and Client Teams" by Gary Kleinman, Dan Palmon, and Picheng Lee [KPL03].

References

[Aal92] W.M.P. van der Aalst, *Timed coloured Petri nets and their application to logistics*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1992.

[AB02] Alan S. Abrahams and Jean M. Bacon, *A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts*, Group Decision and Negotiation **11** (2002), no. 6, 487–524.

[Abr02] Alan S. Abrahams, *Developing and executing electronic commerce applications with occurrences*, Ph.D. thesis, University of Cambridge Computer Laboratory, 2002.

[AH94] K. A. Andersen and J. N. Hooker, *Bayesian logic*, Decision Support Systems **11** (1994), no. 2, 191–210.

[BCK00] Hemant K. Bhargava, Vidyanand Choudhary, and Ramayya Krishnan, *Pricing and product design: Intermediary strategies in an electronic market*, International Journal of Electronic Commerce **5** (2000), no. 1, 37–56.

[BDLT00] Roger W.H. Bons, Frank Dignum, Ronald M. Lee, and Yao-Hua Tan, *A formal analysis of auditing principles for electronic trade procedures*, International Journal of Electronic Commerce **5** (2000), no. 1, 57–82.

[Bha90] Hemant Kumar Bhargava, *A logic model for model management: An embedded languages approach*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1990, Available as a technical report from the Department of Operations and Information Management.

[BI94] Michael Bieber and Thomás Isakowitz, *Text editing and beyond: A study in logic modeling*, Decision Support Systems **11** (1994), no. 2, 219–240.

[BKM97] Hemant K. Bhargava, Ramayya Krishnan, and Rudolf Müller, *Electronic commerce in decision technologies: A business cycle analysis*, International Journal of Electronic Commerce **1** (1997), no. 4, 109–128.

[BL88] Marvin Belzer and Barry Loewer, *A conditional logic for defeasible beliefs*, Decision Support Systems **4** (1988), no. 1, 129–142.

[Bla90] Robert W. Blanning, *The sensitivity properties of hierarchical logic-based models*, Decision Support Systems **6** (1990), no. 2, 89–97.

[Bla94] ———, *A relational algebra for propositional logic*, Decision Support Systems **11** (1994), no. 2, 211–218.

[Bon97] Roger W.H. Bons, *Designing trustworthy trade procedures for open electronic commerce*, Ph.D. thesis, EURIDIS at Erasmus University, Rotterdam, The Netherlands, September 1997.

[Cau94] Robert L. Causey, *EVID: A system for interactive defeasible reasoning*, Decision Support Systems **11** (1994), no. 2, 103–131.

[Che92] K. T. Chen, *Schematic evaluation of internal accounting control systems*, Ph.D. thesis, University of Texas at Austin, Austin, Texas, 1992.

[CK97] Ivo Cathomen and Stefan Klein, *The development of FEDI in Switzerland: A life-cycle approach*, International Journal of Electronic Commerce **1** (1997), no. 4, 129–145.

[Cov97] Michael A. Covington, *On designing a language for electronic commerce*, International Journal of Electronic Commerce **1** (1997), no. 4, 31–48.

[Cov98] ———, *Speech acts, electronic commerce, and KQML*, Decision Support Systems **22** (1998), no. 3, 203–211.

[Das99] Aspassia Daskalopulu, *Logic-based tools for the analysis and representation of legal contracts*, Ph.D. thesis, Department of Computing, Imperial College, University of London, 1999.

[DDM02] Aspassia Daskalopulu, Theo Dimitrakos, and Tom Maibaum, *Evidence-based electronic contract performance monitoring*, Group Decision and Negotiation **11** (2002), no. 6, 469–485.

[Dew92] S. D. Dewitz, *Contracting on a performative network: Using information technology as a legal intermediary*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[DK99] Frank Dignum and Ruurd Kuiper, *Specifying deadlines with continuous time using deontic and temporal logic*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 67–86.

[DRL94] Sandra K. Dewitz, Young Ryu, and Ronald M. Lee, *Defeasible reasoning in law*, Decision Support Systems **11** (1994), no. 2, 133–155, Typographical error in publication. The first author is Sandra D. Dewitz.

[DS02] Aspassia Daskalopulu and Marek Sergot, *Computational aspects of the FLBC framework*, Decision Support Systems **33** (2002), no. 3, 267–290.

[Fan03] Christina Fang, *Strategy as valuation*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 2003.

[FKP+02] Christina Fang, Steven O. Kimbrough, Stefano Pace, Annapurna Valluri, and Zhiqiang Zheng, *On adaptive emergence of trust behavior in the game of stag hunt*, Group Decision and Negotiation **11** (November 2002), no. 6, 449–467.

[GKS97] Georg Geyer, Christoph Kuhn, and Beat Schmid, *Designing a market for quantitative knowledge*, International Journal of Electronic Commerce **1** (1997), no. 4, 89–108.

[GVN94] P. Geerts, D. Vermeir, and D. Nute, *Ordered logic: Defeasible reasoning for multiple agents*, Decision Support Systems **11** (1994), no. 2, 157–190.

[Her96] Henning Herrestad, *Formal theories of rights*, Ph.D. thesis, University of Oslo, Oslo, Norway, 1996, Available as ISBN 82-7833-015-0, Karnovs Forlag, publisher.

[HK98] Gary H. Hua and Steven O. Kimbrough, *On hypermedia-based argumentation decision support systems*, Decision Support Systems **22** (1998), no. 3, 259–275.

[Hoo88] J. N. Hooker, *A quantitative approach to logical inference*, Decision Support Systems **4** (1988), no. 1, 45–69.

[Hua95] Hua Hua, *Theory and applications of argumentation support systems*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1995, Available as a technical report from the Department of Operations and Information Management.

[Jer88] Robert G. Jeroslow, *Spatial imbedding for linear and for logic structures*, Decision Support Systems **4** (1988), no. 1, 71–86.

[Jon02] Andrew J.I. Jones, *On the concept of trust*, Decision Support Systems **33** (2002), no. 3, 225–232.

[KA88] Steven O. Kimbrough and Fred Adams, *Why nonmonotonic logic?*, Decision Support Systems **4** (1988), 111–127.

[Kim99] Steven O. Kimbrough, *Formal language for business communication: Sketch of a basic theory*, International Journal of Electronic Commerce **3** (Winter 1998–99), no. 2, 23–44.

[Kim80] ———, *The concepts of fitness and selection in evolutionary biology*, Journal of Social and Biological Structures **3** (1980), 149–170.

[Kim90] ———, *On representation schemes for promising electronically*, Decision Support Systems **6** (1990), no. 2, 99–121.

[Kim03] ———, *Computational modeling and explanation: Opportunities for the information and management sciences*, Computational Modeling and Problem Solving in the Networked World: Interfaces in Computing and Optimization (Hemant K. Bhargava and Nong Ye, eds.), Operations Research/Computer Science Interfaces Series, Kluwer, Boston, MA, 2003, pp. 31–57.

[KL88] Steven O. Kimbrough and Ronald M. Lee, *Logic modeling: A tool for management science*, Decision Support Systems **4** (1988), no. 1, 3–16.

[KL97] ———, *Formal aspects of electronic commerce: Research issues and challenges*, International Journal of Electronic Commerce **1** (1997), no. 4, 11–30.

[KM93] Steven O. Kimbrough and Scott A. Moore, *On obligation, time, and defeasibility in systems for electronic commerce*, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III, Information Systems: DSS/Knowledge-Based Systems (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1993, pp. 493–502.

[KPL03] Gary Kleinman, Dan Palmon, and Picheng Lee, *The effects of personal and group level factors on the outcomes of simulated auditor and client teams*, Group Decision and Negotiation **11** (2003), no. 1, 57–84.

[Kri90] Ramayya Krishnan, *A logic modeling language for automated model construction*, Decision Support Systems **6** (1990), no. 2, 123–152.

[Kro97] Cristen Krogh, *Normative structures in natural and artificial systems*, Ph.D. thesis, University of Oslo, Oslo, Norway, 1997, Available at http://www.uio.no/∼krogh/ documents/papers/complex97.ps.

[KT00] Steven O. Kimbrough and Yao-Hua Tan, *On lean messaging with unfolding and unwrapping for electronic commerce*, International Journal of Electronic Commerce **5** (2000), no. 1, 83–108.

[KWZ02] Steven O. Kimbrough, D. J. Wu, and Fang Zhong, *Computers play the beer game: Can artificial agents manage supply chains?*, Decision Support Systems **33** (2002), no. 3, 323–333.

[Lee80] Ronald M. Lee, *CANDID: a logical calculus for describing financial contracts*, Ph.D. thesis, The Wharton School, University of Pennsylvania, Philadelphia, PA, 1980, Available as WP-80-06-02, Department of Operations and Information Management (née Decision Sciences).

[Lee88] ———, *A logic model for electronic contracting*, Decision Support Systems **4** (1988), no. 1, 27–44.

[Lee99] ———, *Distributed electronic trade scenarios: Representation, design, prototyping*, International Journal of Electronic Commerce **3** (1999), no. 2, 105–136.

[Lee02] ———, *Automated generation of electronic procedures: Procedure constraint grammars*, Decision Support Systems **33** (2002), no. 3, 291–308.

[Lom99] Alessio R. Lomuscio, *Information sharing among ideal agents*, Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, UK, February 1999.

[LS99] Markus A. Lindemann and Beat F. Schmid, *Framework for specifying, building, and operating electronic markets*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 7–22.

[LWJ03] Alessio R. Lomuscio, Michael Wooldridge, and Nicholas R. Jennings, *A classification scheme for negotiation in electronic commerce*, Group Decision and Negotiation **11** (2003), no. 1, 31–56.

[Moo93] Scott A. Moore, *Saying and doing: Uses of formal languages in the conduct of business*, Ph.D. thesis, University of Pennsylvania, The Wharton School, Philadelphia, PA, 19104, USA, December 1993.

[Moo98] ———, *Categorizing automated messages*, Decision Support Systems **22** (1998), no. 3, 213–241.

[Moo00] ———, *KQML and FLBC: Contrasting agent communication languages*, International Journal of Electronic Commerce **5** (2000), no. 1, 109–124.

[NE98] Donald Nute and Katrin Erk, *Defeasible logic graphs: I. theory*, Decision Support Systems **22** (1998), no. 3, 277–293.

[NHH98] Donald Nute, Zachary Hunter, and Christopher Henderson, *Defeasible logic graphs: II. implementation*, Decision Support Systems **22** (1998), no. 3, 295–306.

[NMB90] Donald Nute, Robert I. Mann, and Betty F. Brewer, *Controlling expert system recommendations with defeasible logic*, Decision Support Systems **6** (1990), no. 2, 153–164.

[Nut88] Donald Nute, *Defeasible reasoning and decision support systems*, Decision Support Systems **4** (1988), no. 1, 97–110.

[Oli96] Jim R. Oliver, *On artificial agents for negotiation in electronic commerce*, Ph.D. thesis, University of Pennsylvania, The Wharton School, Department of Operations and Information Management, Philadelphia, PA, April 1996.

[Oli97] ———, *Artificial agents learn policies for multi-issue negotiation*, International Journal of Electronic Commerce **1** (1997), no. 4, 49–88.

[Ong92] Kay Liang Ong, *A formal model for maintaining consistency of evolving bureaucratic policies: A logical and abductive approach*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[PT02] Balaji Padmanabhan and Alexander Tuzhilin, *Knowledge refinement based on the discovery of unexpected patterns in data mining*, Decision Support Systems **33** (2002), no. 3, 309–321.

[Roe92] Stephen F. Roehrig, *Probabilistic and defeasible reasoning using extended path analysis*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1992, Available as a technical report from the Department of Operations and Information Management.

[Ryu92] Young U. Ryu, *A formal representation of normative systems: A defeasible deontic reasoning approach*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[Ryu98] ———, *A logic-based modeling of resource consumption and production*, Decision Support Systems **22** (1998), no. 3, 243–257.

[Sco88] Peter C. Scott, *Requirements analysis assisted by logic modeling*, Decision Support Systems **4** (1988), no. 1, 17–25.

[SF90] Shimon Schocken and Tim Finin, *Meta-interpreters for rule-based inference under uncertainty*, Decision Support Systems **6** (1990), no. 2, 165–181.

[TT99] Yao-Hua Tan and Walter Thoen, *A logical model of directed obligations and permissions to support electronic contracting*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 87–104.

[TT02] ———, *Formal aspects of a generic model of trust for electronic commerce*, Decision Support Systems **33** (2002), no. 3, 233–246.

[vR96] Victor Emil van Reijswoud, *The structure of business communication: Theory, model and application*, Ph.D. thesis, Technische Universiteit Delft, Delft, The Netherlands, June 1996, ISBN: 90-9009439-3. Address: Victor E. van Reijswoud, Juffrouw Idastraat 1, 2513 BE Den Haag, The Netherlands.

[Wid88] George R. Widmeyer, *Logic modeling with partially ordered preferences*, Decision Support Systems **4** (1988), no. 1, 87–95.

[Wid90] ———, *Reasoning with preferences and values*, Decision Support Systems **6** (1990), no. 2, 183–191.

[Win97] Hung Wing, *Managing complex, open, web-deployable trade objects*, Ph.D. thesis, University of Queensland, Department of Computer Science and Electrical Engineering, The University of Queensland, Brisbane Qld 4072, Australia, December 1997.

[WS02] D. J. Wu and Yanjun Sun, *Cooperation in multi-agent bidding*, Decision Support Systems **33** (2002), no. 3, 335–347.

[WSdMD03] Hans Weigand, Mareike Schoop, Aldo de Moor, and Frank Dignum, *B2b negotiation support: The need for a communication perspective*, Group Decision and Negotiation **11** (2003), no. 1, 3–29.

[Wu97] D. J. Wu, *Using genetic algorithms to determine near-optimal pricing, investment and operating strategies in the electric power industry*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA, 1997.

[Wu00] ———, *Artificial agents for discovering business strategies for network industries*, International Journal of Electronic Commerce **5** (2000), no. 1, 9–36.

[Wv99] Hans Weigand and Willem-Jan van den Heuvel, *Meta-patterns for electronic commerce transactions based on the formal langauge for business communication (FLBC)*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 45–66.

[Wv02] ———, *Cross-organizational workflow integration using contracts*, Decision Support Systems **33** (2002), no. 3, 247–265.

[ZKW02] Fang Zhong, Steven O. Kimbrough, and D. J. Wu, *Cooperative agent systems: Artificial agents play the ultimatum game*, Journal of Group Decision and Negotiation **11** (November 2002), no. 6, 433–447.

Part I

Representation: Objects, Processes & Policies

Practical Contract Storage, Checking, and Enforcement for Business Process Automation

Alan Abrahams[1], David Eyers[2], and Jean Bacon[3]

[1] University of Cambridge, Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK,
asa28@wharton.upenn.edu
[2] University of Cambridge, Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK,
David.Eyers@cl.cam.ac.uk
[3] University of Cambridge, Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK,
Jean.Bacon@cl.cam.ac.uk

Abstract. We show how Kimbrough's Disquotation Theory, a formal theory about sentences that embed propositional content, can be profitably applied to the creation of computational environments for monitoring and enforcing electronic commerce contracts using pervasive, mainstream industrial technologies such as Java and relational databases. We examine the notion of an occurrence and provide a structural representation of this abstraction. We show how contractual provisions - obligations, permissions, prohibitions, and powers - can be stored, monitored, and enforced. Detailed examples illustrate how a query coverage-determination mechanism can be used to check inter-organizational contractual provisions against internal policies and external legislation for dynamic conflicts. The work presented here demonstrates that an extended version of Kimbrough's theory presents a novel and promising means of storing interrogable and executable specifications for e-commerce workflow applications.

1 Introduction

Implicit and explicit contracts are an important mechanism for coordinating the activities of an organization. In their contracts, organizations express which states of affairs are desirable and undesirable. They specify which legal states of affairs are achievable, and how, and which are not. In each case, a subjective attitude towards propositional content is stated: the state of affairs is obliged, forbidden, permitted, obtains in law, or does not, according to some clause. We demonstrate here that representing and reasoning about subjective, propositional content has applications in the automation of commerce.

We favor a broadly declarative approach. A procedural execution style hardwires process sequences and requires a complex, manual search-and-fix

strategy each time a business's contracts change. The obligations, permissions, and powers expressed in contracts are not explicit in procedural code, being lost in line-by-line invocations. Traditional imperative languages, which provide no direct representation of rights and duties of parties, therefore introduce an impedance mismatch between the contracts that specify the desired system behavior, and the software that implements these requirements. In conventional imperative language approaches, rules and procedures are obscured in computer code and not directly accessible or modifiable; instead of reproducing rules in procedure manuals, memoranda, and dispersed across computer code, a centralized rule base should be used to ensure locality of update and avoid update inconsistencies [Lee88a]. Our goal is to store explicit representations of the obligations, permissions, and powers inherent in contracts in order that we may use these stored contractual provisions to executably specify workflow applications. Our approach to the representation of rights and duties has been informed by the vast literature on deontic logic, and in particular by Kimbrough's Disquotation Theory. In our software realization, we adopt a declarative, event-driven approach, where each contract provision is explicitly stored and monitored, and software and human components may consult the provisions to determine what currently holds, and what to do next.

Kimbrough's Disquotation Theory [Kim01] is a nascent formal theory that can be used for representing and reasoning about sentences that embed propositional content, such as those commonly found in electronic commerce contracts. We find ourselves in broad agreement with Kimbrough's theory and wish to offer many detailed amendments and extensions. We seek to describe a software implementation of Kimbrough's theory that provides an environment in which we can examine practical application development cases. Our contribution is to move beyond a purely logical view and towards a system that can be implemented and used. Applications and practice very properly have roles in informing the development of the theory. Many have preceded us in the theory department. A vast philosophical literature exists on deontic logic,[1] speech acts [Aus62,SV85], and event semantics.[2] Our goal here is to present a practical approach aimed at mimicking the devices of a particular theory and establishing its plausibility through testing with real applications. We hope to contribute to the theory through an understanding of how to implement the promising ideas. We explain the data structure and software features implemented in our prototype development and execution environment, EDEE, which is capable of representing, storing, and enforcing electronic commerce contracts over diverse platforms using occurrence stores. Our software includes a novel continuous query mechanism for determining which stored provisions apply to a new occurrence.

[1] See [MW93], [And58], [And62], [Das99], [Mak86], [Lin77], [Lee88a], and [PS97] for starters.

[2] See, e.g., [Dav80], [Ben88], [Par90], [JM00], and [PV00].

We begin with a brief application scenario (§3), and an overview of our approach to storing and executing contractual provisions (§4). Section 5 then describes what is meant by an occurrence and demonstrates how an occurrence may be used to store a workflow event, such as a payment, or an association state, such as 'being a supplier for'. In Section 6, we give a brief review of Kimbrough's Disquotation Theory. Section 7 explains how Kimbrough's notions may be implemented in software: we show that stored queries can be used as the implementation mechanism of Kimbrough's quotation construct that is used for bracketing propositions within deontic statements. Our representations of common contractual provisions such as prohibitions, permissions, powers and liabilities, and obligations are shown (§7.2 - §7.5), and we demonstrate how the use of functions provides a means of determining the necessary deontic and legal consequences of particular provisions. We discuss a number of detection, intervention, and prevention strategies that may be employed to find or avoid violations (§8 and §9). Section 10 reviews the features of our software prototype and explains the novel mechanism used for finding conflicting provisions. We conclude with a comparison to related work.

2 Contributions

In this paper, our investigations into event semantics lead to the definition of a database schema for the representation of these variable-attribute occurrences (see also [AB01b]), paving the way to interrogation and execution of stored e-commerce application specifications. Our EDEE prototype provides a platform-independent active wrapper (see also [AB01d]), which allows us to record, reason about, and enact contractual provisions.[3] We demonstrate a novel query storage and coverage determination mechanism, which allows contract performance monitoring and facilitates dynamic consistency checking of contracts against policies.[4]

In our earlier work, a new model of the life and times of identified and situated norm instances was proposed.[5] The model is used in our contract-driven and legislation-aware workflow automation approach, to support conflict resolution [AEB02a]. Our previous work also provided a set of guidance rules, which could be employed by an analyst to expose salient occurrences in English language user requirements documents, such as business contracts, policies, and legislation.[6]

The work described here contributes to the understanding of the representation and implementation of contracts, policies, and legal requirements, for business process automation. The use of a database to record history in an

[3] See also [AB01c,AB02c].
[4] See also [Abr02,AEB02b,AEB02c].
[5] [AB02a,AK02]
[6] [AB00,AB01a,Abr02]

integrated form enables querying of state and history, which is essential to the abstract modelling of processes at the business level. We propose a significant attempt to advance the state of the art in the capturing of user requirements and their mapping to computer systems functions such as access control, enactment, and audit. This research may stimulate new approaches to delivering security and integrity to business processes, and has the potential to feed into developer tools based on policy specification and enforcement. The novel development style described could have a significant impact on the automation of activities in commercial organisations and would be an important contribution to workflow management in e-business. Progress towards the creation of an integrated methodology and environment for provision identification, representation, storage, conflict detection and resolution, monitoring, and enforcement presents a significant engineering challenge, and constitutes the main contribution of this work.

3 Application Scenario

To clarify the problem we are seeking to address, consider the following application scenario, which we will return to throughout the paper.

SkyHi Builders is a constructor of small office buildings. Steelman's Warehouse is a supplier of high-grade steel. SkyHi, having recently won a tender to build a new office block, enters into a contract with Steelman's Warehouse. An excerpt appears as follows:

Contract between SkyHi Builders and Steelman's Warehouse

...

SkyHi Builders must pay Steelman's Warehouse £25,000 before 1st September 2000 (Clause C.1)

...

In addition, SkyHi has the following internal organizational policies:

SkyHi Risk Management Procedures

...

Clerks may not buy steel. (Clause P.1)

Employees older than 25 may buy steel. (Clause P.2)

Payments of more than £10,000 to suppliers are prohibited. (Clause P.3)

...

And SkyHi finds themselves subject to the following provisions of legislation:

Commercial Trade Act

...

Performance of all actions fitting the description of obliged actions counts as filling that obligation.	(Clause L.1)
Failure to perform all actions fitting the description of obliged actions by the deadline counts as violating the obligation.	(Clause L.2)
Following successful instigation of the prescribed procedure for claiming compensation, damages for violation of an obligation must be paid, by the liable party, to the party entitled to compensation.	(Clause L.3)

...

SkyHi wishes to store the provisions of their contracts and internal business policies, in addition to the legal regulations to which they are subject, in a database, so that the provisions can be used to guide the behavior of their (computer and human-activity) systems. Scripting the system with procedural code is not an option: the sequence of SkyHi's business processes is not static and they do not wish to employ a programmer to sift through and change procedural code to reflect the frequent alterations in contracts, policies, and regulations. SkyHi would like to the human and software components in their system to consult the database in order to determine what to do next in the light of a dynamically changing set of provisions. They need to use the provisions to ascertain the state-of-affairs that obtains at the present moment. They need to store a history of events and states, so that they know what occurrences have happened over time. They want to know, for instance, what consequential obligations and legal powers resulted, and whether these obligations were fulfilled or powers were exercised. Further, they need to be able to assess whether a given activity is permitted, to determine which historical occurrences where prohibited, and what violations ensued.

4 Overview

We view a **contract**, **policy**, or **regulation**, as a *set of provisions*. A **provision** specifies an obligation (§7.5), permission (§7.3), prohibition (§7.2), or power (§7.4). Provisions are described in clauses. Broadly, provisions represent rights and duties. We wish to represent and store these provisions and their contents as records in a database.

Figure 4 presents the general architecture of our EDEE environment. Contractual provisions are stored in the database, along with workflow occurrences. When business and environmental occurrences are inserted into the database, an active database layer consults the stored provisions to determine which provisions cover those occurrences, and consequently, which obli-

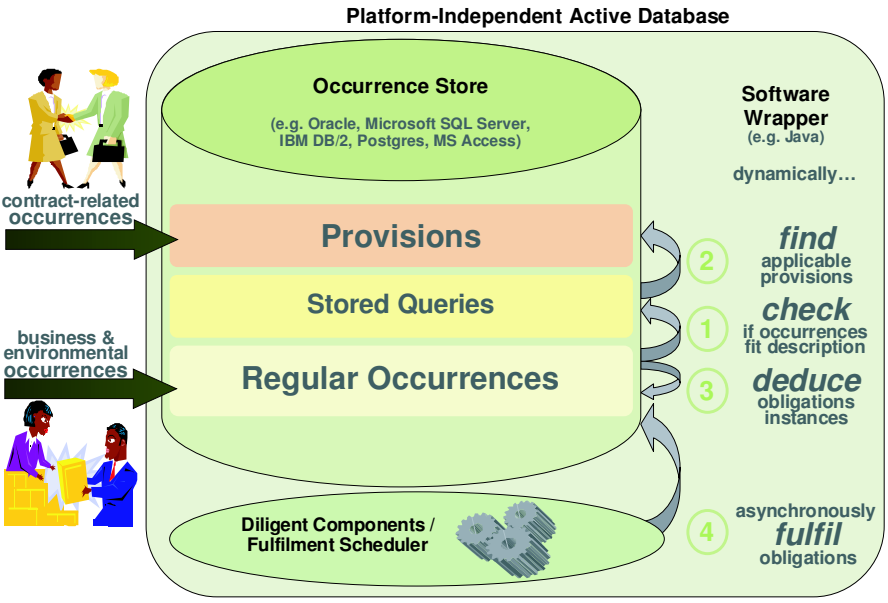


Fig. 1. General architecture of EDEE

gations, permissions, prohibitions, and powers apply. A separate fulfillment scheduler consults the list of active obligations and permissions, and attempts to fulfil the relevant provisions. The database is not intended as a general e-marketplace, but rather as a repository of the current contracts and policies of a single company. Together these contracts and policies represent the workflow system specification for that company. Our system is intended for inter-organizational contracts, internal policies within an organization, and external legislation, since all specify the norms that govern the organization and guide execution. Execution components use the database to ascertain what they are obliged to do: that is, what procedures to enact. Users may query the database to ascertain what legal relations have obtained as a result of past occurrences.

What is meant by 'contract enforcement' in this context? Are norms that the system infers as applicable at a given point in time intended to be *imposed* on the business process, with no possibility of violation, or are parties merely to be informed that such norms apply and have the *freedom* to violate them? The answer depends on the type of norm. In the case of *descriptive* norms, which govern construal of a situation, the construal is forced by the system. Consider, in particular, a norm such as 'if Steelman's fails to deliver by 10pm then, according to clause 2.4, Steelman's is *obliged* to refund the purchase price'. Here, the construal that a particular obligation arises as a result of a failure to deliver is forced by the rule specification. There can be no argument that, in terms of that rule taken on its own, such an occurrence

(an *obligation according to clause 2.4*) has come about. The case of *prescriptive* norms, which prescribe behavior, is somewhat different. Our assumption is that a party is free to observe or violate such norms, but bound to accept the legal consequences of their actions. For example, for the obligation according to clause 2.4 above, Steelman's is free to refund the purchase price or not; but if they don't they are bound to be construed as being in violation of the obligation to refund. Existing approaches to policy specification (see for example [D$^+$01,DDLS01]), primarily targeted at network management, assume that components lack free will and execute all their obligations immediately and without delay. In commercial environments though, agents typically have more flexibility in scheduling the fulfillment of their obligations: immediate enacting is one possible option which may or may not minimize overall response time or maximize profitability. With commercial obligations it is more often the case that agents can fulfill an obligation by acting at any time before a deadline or within an interval. Furthermore, while we agree that in the ideal world all obligations are fulfilled, real world practicalities such as resource limitations, conflicting obligations, unpredictable environmental forces, and free agents make this ideal unattainable; the possibility of deviation from prescribed behavior should not be overlooked [DDM01]. The intention of obligations is, we believe, not only to specify what should ideally take place, but also what happens when the ideal cannot be met, or when things go wrong. We must qualify our assumption of free will though, since we do not mean to imply that the software components are simply free to ignore obligations. Rather, our architecture requires that software components are *diligent*: they consult the database to ascertain what obligations currently apply, or are notified of such by the active database, and attempt to fulfil them within their resource constraints. Similarly for human components. When they fail, or indeed when they succeed, they are forced to accept the construals of the database as to what obligations come into play, and as to what occurrences are deemed to have occurred as a result of their action (or inaction).

Unlike Standard Deontic Logic, which presumes that being-obliged to do that which is prohibited is a logical contradiction (obligation and prohibition are treated as operators), we assume that prohibitions and obligations are independent entities (variables that are quantified over), and conflicting norms can exist. An agent may find itself violating a prohibition in order to fulfil an obligation, or vice versa. Entirely avoiding such conflicts is not our intention since we believe, in a world of multiple norm-givers, conflicts and consequent trade-offs are inevitable. Indeed, even a single norm promulgated by a single party can result in conflict. Consider Hansson and Makinson's example [HBC97] of a single imperative "a doctor must visit heart-attack victims immediately" which can produce conflicts: if John and Jeff, living at remote locations in the rural outback, both suffer heart-attacks simultaneously, the local doctor has two conflicting obligations and must decide who to

visit first, since, given his resource constraints, he cannot satisfy both "I must visit John immediately" and "I must visit Jeff immediately". Hansson and Makinson's view is that, since conflicts cannot exist, one of these norms must be blocked (restrained from the output set of norms) to eliminate the conflict. Our view is that it is undeniable, prima facie, that both obligations exists. It may be that one is violated if the doctor visits only one of the patients. More likely though, one of the obligations is voided by a separate principle of fairness which effectively specifies that a violation in this case is forgivable. Other principles may exist which specify another obligation, to visit the second patient within a more lenient time interval, that can be fulfilled by the doctor. Some ordering criteria must specify which patient is 'second'. In an e-commerce scenario, Hansson and Makinson's example might become "Suppliers are obliged to dispatch books within 24 hours of order". In the presence of one or more orders, the various obligations that this generates may become unfulfillable as a result of any number of resource limitations: inventory, human resources, monetary resources, machinery, or time. Our intention is to accept that conflicts are common, and be able to specify conflicting provisions so that we may detect such conflicts and flag them to the norm-giver. We deal with conflict resolution in [Abr02,AB02a,AEB02a].

We now turn to an explanation of what is meant by the notion of an 'occurrence'.

5 Occurrences

We treat an 'occurrence' as being a instance of a specific relationship type or association type that exists between entities, at a moment in time or over an interval in time. For instance, we would treat *buying*, *owning*, *approving*, *being-obliged*, and *being-prohibited* as occurrences. Each occurrence has role-players acting in a role in the occurrence: an occurrence of 'buying', typically has at least participants in the roles *buyer*, *seller*, *sold item*, and *purchase price*. An occurrence may be an event, a state, or indeed even a process. Consider an *event* of 'supplying' which is instantaneous, and has as participants a supplier, a supplied party, and an item supplied at that instant. In contrast, a *state* of 'being supplier' has (perhaps unspecified) duration, where the supplier and supplied party have an association for the whole duration of the state. And a *process* of supplying does not imply that what is being supplied is ever supplied. When referring to an occurrence of 'supplying' we should generally clarify which semantics (event, state, or process) we mean, and also which sense of the word we intend. Each occurrence has a single type, and, an occurrence type (such as 'buying') may have multiple occurrence instances. Different occurrences may overlap in time: John may be involved in a process of buying a fridge, while Jeff is involved in a process of buying a car. Indeed, if both processes are brought to a successful conclusion by the single fall of a hammer in a clearance sale, then the occurrence of the instantaneous events

of buying could occur simultaneously as well. For this reason, occurrence times are insufficient to identify occurrences, and we require instead unique identifiers.

We have chosen the term 'occurrence' since the term 'event', used in the philosophical literature[7] to describe a momentary or prolonged state of affairs, is typically connoted as being instantaneous in the active database and systems programming literature [PD94]. Furthermore, we do not wish to imply that our notion shares any of the philosophical subtleties of various uses of the term 'event' in the literature. For instance, Bennett [Ben88], controversially (though he is not alone is his sentiment) argues that John's crossing the Channel and John's swimming the Channel are the same 'event', whereas we treat them as separate occurrences. Kimbrough (personal communication) argues that an obligation state, ought(e), can be the same as a violation state, violating(e). Our contention is that each is an independent entity: an occurrence, being_obliged1, and an occurrence, violating1, where the participant in the role 'violated' in violating1 is the obligation being_obliged1. Nevertheless, our notion of an 'occurrence' most closely resembles Parsons's notion of an 'eventuality' (an event, state, or process), which Kimbrough [Kim01,Kim02] has applied to deontic reasoning.

The question arises as to why we treat occurrences of events, states, and processes uniformly in certain respects. Our contention is that all indicate a temporal relatedness between participants in various roles, and the abstraction of a notion of an 'occurrence' is therefore useful. Policies such as 'Bob is forbidden to *buy* X' or 'Bob is forbidden to *own* X' speak of occurrences of *buying* and *owning* in pragmatically similar ways even though the former seems to be an occurrence of an event and the latter an occurrence of a state. The difference, we argue, is only significant in certain respects, and for our purposes we wish to exploit the similarity in order to uniformly monitor for the occurrence of events, states, and processes over time. Furthermore, we can provide a uniform set of facilities for querying occurrences, and their role-holders, without necessarily having to explicitly decide in each case whether the occurrence is instantaneous or prolonged.

Take the scenario where Steelman's Warehouse is a supplier for SkyHi Builders, and SkyHi Builders has paid £25,000 to Steelman's Warehouse for a specific shipment. Ignore for the moment any contractual relationships (contract-related occurrences), which we return to later, and consider only the simple occurrence of being a supplier, and an occurrence of paying. Let being_supplier1 and paying1 be Skolem constants that name occurrence instances of types *being a supplier* and *paying* respectively. We might add allocating1 to denote that the payment was *allocated*, using the First-In First-Out basis, to a delivery, and occurring1 to indicate that the payment *occurred* on 15$^{\text{th}}$ August 2001. A basic occurrence-role-participant tabular schema can be employed for storing occurrences: in each row, we store the occurrence iden-

[7] E.g., [Dav80,Par90,PV00].

Table 1. A tabular schema for storing various occurrences

Commentary	**Occurrence**	**Role**	**Participant**
Represent Steelman's *being a supplier* for SkyHi	being_supplier1	supplier	Steelman's
		supplied	SkyHi
...			
According to Clause 1, SkyHi *paid* £25,000 to Steelman's	paying1	payer	SkyHi
		paid-amount	£25,000
		payee	Steelman's
		isAccordingTo	Clause 1
The payment was *allocated* to a previous shipment	allocating1	allocated	paying1
		allocatedTo	shipment1
		allocationBasis	FIFO
		isAccordingTo	Clause 2
The payment *occurred* on 15 August 2001	occurring1	occurred	paying1
		occurredOn	15 August 2001

tifier, a participant and its role in the occurrence. Table 1 depicts a tabular representation of the above-mentioned occurrences. For readability we have included values like Steelmans in our tables instead of foreign key references. Similarly we show occurrence primary keys in forms such as being_supplier1, instead of foreign key references into a table describing the occurrence type *being_supplier*. Finally we omit repeated key values in adjacent rows.

We should comment at this point that an occurrence store ought to capture subjective propositions. By this we mean the fact that propositions ought not to be believed to be true until a principled reasoning process that accounts for conflicts and contextual information has been undergone. This means that the parsing of the sentence 'SkyHi Builders paid £25,000 to Steelman's Warehouse' should *not* lead to the direct representation of the objective fact that 'authoritatively, Steelman's Warehouse was paid', since this would prematurely and recklessly presume the truth of the proposition; rather the parsing of the sentence should lead only to the representation of 'a particular clause (utterance) *asserts* that Steelman's Warehouse was paid' or, synonymously, '*according to the utterance*, Steelman's Warehouse was paid'. Thus, while there is clearly contention, there should be no logical contradiction in 'Clause2 *says* Steelman's Warehouse was paid for shipment1' and 'Clause3 *says* Steelman's Warehouse was not paid for shipment1' since both clauses may have been said and indeed, under different assignments of payments to performances, both clauses may indeed be subjectively true. The contention between the opinions can be resolved by a principled (i.e., rule-based) decision as to which utterance to *accept* at a given time (in a given circumstance). Making each assertion relative to a clause (utterance or provision) is a means of embedding subjectivism in the representation; in Table 1 we have achieved this by tagging each posited occurrence with a clause (utterance identifier) which asserts its existence. This subjectivism provides us with the ability to

make rational choices as to which uttered *content* is believed over time, and allows belief revision through choice of alternative subjective statements.

The storage schema depicted bears some resemblance to the graph-based binary-relational model used by the functional database language Hydra [AK96], and shares some of the same useful properties. Ayres and King comment that conventional relational and object-oriented database systems are unable to support associational queries corresponding to questions such as 'what is the relationship between Sandra and Mike?'. Hydra, in contrast, provides associational primitives, which return the list of functions or inverse-functions that relate two particular items. Our occurrence-centric representation provides similar functionality advantages, allowing the framing of queries such as

```
(select occurrences where Steelman's Warehouse participates)
intersection (select occurrences where SkyHi participates)
```

This would return paying1 and being_supplier1 in the example. This style of associational query is especially useful in commerce applications for determining legal relationships between parties. For instance, an associational query may be framed to determine the history of interactions between two parties, or indeed to determine what obligations (occurrences of being-obliged) bind two parties.

Kimbrough's ESΘ theory [Kim98a,Kim01], based on proposals by Parsons [Par90], represents events and states using event variables and thematic roles. An occurrence of SkyHi Builders paying £25,000 to Steelman's Warehouse for a particular shipment would be formally represented as:

$$\begin{aligned}\exists e(\text{paying}(e) &\wedge \text{Subject}(e, SkyHiBuilders)\\ &\wedge \text{DirectObject}(e, £25000)\\ &\wedge \text{IndirectObject}(e, Steelman's Warehouse)\\ &\wedge \text{Sake}(e, e\prime)\\ &\wedge \text{shipment}(e\prime))\end{aligned} \tag{1}$$

The ways in which our 'occurrences' differ from Kimbrough's events and states could be summarized as:

- We see occurrences as being of a single type, implying obligation states and violation states are distinct entities
- We choose to use domain- or application-specific roles - so-called 'deep roles' [JM00] - rather than generic thematic roles (such as **agent**, **theme**, etc.) or grammatical syntax markers (such as **subject**, **object**, etc.), in the representation of occurrences. While generic thematic roles - such as **agent**, **patient**, **instrument**, **source**, and **destination** - are often helpful and are commonly used in knowledge representation in artificial intelligence [All95,Sow00], they have been criticized. Davis [Dav96], for

instance, argues that the thematic role of a participant in an event occurrence may be difficult to determine or ambiguous. Reliance solely on thematic roles may therefore be problematic as it has been argued that thematic roles do not always uniquely identify participants in a given role in a commercial occurrence. Citing Jackendoff, Fillmore, Gawron, and Croft, Davis explains that in a commercial event such as 'buying' both the buyer and the seller can be construed to be in the **agent** thematic role of the event occurrence, as the buyer acts to supply money and the seller acts to supply goods. Using domain-specific roles (such as **buyer**, **seller**, etc.) allows us to avoid this vagueness. An alternative to using thematic roles is to use grammatical syntax markers such as **subject** and **object**. Kimbrough, acknowledging that it is almost certainly preferable to employ semantic markers instead, uses these for illustration purposes. The problem with grammatical syntax markers is that they are vague when it comes to semantics: consider that, in 'John opened the door' (active or causitive reading) and 'the door opened' (passive or inchoative reading), the door occupies different syntactic positions - **object** and **subject** respectively - even though its semantic role as **being opened** stays constant. One of the triumphs of event semantics is semantic stability in its treatment of active and passive sentences [Par90]; a benefit lost by simplifying the illustration with grammatical syntax markers. We prefer to employ domain-specific roles, such as **opener** and **opened**, as these are stable, precise, and simple for an analyst untrained in linguistics to identify and comprehend.

- We employ 'allocating' occurrences in place of Kimbrough's Sake(...) predicate to allow different parties to allocate performances to obligations using different bases.
- To support measures by different parties over time, we employ measuring occurrences [Abr02] instead of Kimbrough's unit() and quantity() predicates [Kim98a].

In the case of simple occurrences, the representation we offer (tuples versus Kimbrough's logical expressions) affords us an implementation mechanism using relational, or simply tabular, data stores. The representation is similar whether the occurrences are stored in a relational database, held in a spreadsheet, or transmitted in a comma-delimited text file. No data transposition is technically necessary, so streaming the data into and out of the database from communication channels can proceed without schema transformation difficulties, and using very simple and efficient parsers. We have not dealt with compression or data clustering techniques, though it seems clear the representation could be stored and accessed more efficiently without loss of information.

In the case of more complex occurrences, which nest queries, the relationship to Kimbrough's Disquotation operator is more removed, but, we believe,

still faithful to the intentions of the logic. The details are described in Sections 7 and 10.

6 Kimbrough's Disquotation Theory

Kimbrough's Disquotation Theory [Kim01] provides a representation of the objects of sentences that embed propositional content. In this section we illustrate briefly Kimbrough's representation of a number of types of sentences with embedded propositions: assertions, permissions, prohibitions, and obligations.

Assertions: Instantiating 'Mary asserts that SkyHi Builders paid £25,000 to Steelman's Warehouse' from Kimbrough's axiom schema for assertions would yield:

$$\begin{aligned} \exists e_1(\text{asserting}(e_1) \wedge \text{Subject}(e_1, Mary) \wedge \text{Object}(e_1, [\phi])) \\ \rightarrow (\text{Veridical}(e_1) \leftrightarrow \phi) \end{aligned} \tag{2}$$

where ϕ represents the predicates from *Formula (1)* above which says, in brief, 'SkyHi Builders was paid'. The brackets '[...]' are special quotation operators that turn their contents into an opaque string; thereby preventing 'Mary asserting that Steelman's Warehouse was paid' from necessarily entailing 'Steelman's Warehouse was paid'. This says that Mary's assertion that SkyHi Builders paid £25,000 to Steelman's Warehouse is veridical (true) if and only if there was an actual occurrence of SkyHi Builders paying £25,000 to Steelman's Warehouse.

Prohibitions: Using Kimbrough's semantics, a prohibition against paying Steelman's Warehouse more than £10,000 could be expressed as:

$$\begin{aligned} \exists e_2(\text{prohibiting}(e_2) \wedge \text{IsAccordingToClause}(e_2, c) \wedge \text{Object}(e_2, [\phi])) \\ \rightarrow (\text{Violated}(e_2, c) \leftrightarrow \phi) \end{aligned} \tag{3}$$

where ϕ is an expression describing an occurrence of Steelman's Warehouse paying more than £10,000. Thus we say the prohibition of SkyHi Builders against paying Steelman's Warehouse is violated, in terms of clause c, if and only if, more than £10,000 is actually paid to Steelman's Warehouse. Note that we have, here and elsewhere, supplanted Kimbrough's original suggestion of an InSystemOfNorms(e,n) predicate, with an IsAccordingToClause(e,c) predicate, to capture the more specific sentence, '... is violated, according to this *clause* (utterance) c, if ...'. We believe this adjustment is necessary as systems of norms (sets of regulations) may contain conflicting provisions that can only be resolved through choice between identified clauses. Therefore, in each provision it is only appropriate to authoritatively state that 'there is violation *in terms of this specific clause*', but not 'there is violation in terms of this system of norms', since other provisions (clauses) in the system of norms

may conflict and override. For instance, the so-called *de minimus* provision of English Law [TB99, p144] may rule that there is no violation in the event that SkyHi paid only £24,499, since the difference is not material.

Permissions: In Kimbrough's semantics, the permission of SkyHi Builders to pay Steelman's Warehouse could be expressed as:

$$\exists e_3(\text{permitting}(e_3) \wedge \text{IsAccordingToClause}(e_3, c) \wedge \text{Object}(e_3, [\phi])) \rightarrow (\neg\text{Violated}(e_3, c)) \tag{4}$$

Kimbrough reads this as 'permissions cannot be violated'. More particularly, we would read this as 'no violations are brought about by permitted occurrences'. A similar interpretation of permission, following Anderson [And58], is provided in Lee [Lee88a], who says that a state of affairs is permitted if and only if it will never be the case that bringing about that state of affairs implies sanctions. We prefer the Kimbrian interpretation, since we see violations as being distinct from sanctions, as the former does not necessarily imply *regress* (penalty), merely *transgress*. Lee's interpretation receives support from Cholvy, Cuppens, and Saurel [CCS97], who argue that it is prejudicial that p is the case if and only if it is necessary that if p occurs then it is obligatory to repair for damage. However, we wish to emphasize that the fact that something is wrong (there is a violation) is distinct from any penalties - 'sanctions' [Lee88a] or 'reparations' [CCS97] - that follow from that violation. That someone has been wronged does not automatically imply an obligation to repair the wrong: there are many cases where no reparations are specified or available, yet the violation is still deemed to have occurred.

Another useful sense of the term 'permission', commonly attributed to Bentham [Lin77], is permission in the sense of 'vested liberty'. A vested liberty is combined with an obligation for others not to prevent the action [Lin77, p17], whereas a naked liberty is combined with freedom for other people to prevent the action in question. Violation (through attempting to prevent a permitted occurrence, or more specifically, through attempting to interfere with a vested liberty) can bring about a set of obligations or prohibitions on a liable party.

Obligations: In Kimbrough's semantics, the obligation of SkyHi Builders to pay Steelman's Warehouse could be expressed as:

$$\exists e_4(\text{ought}(e_4) \wedge \text{IsAccordingToClause}(e_4, c) \wedge \text{Object}(e_4, [\phi])) \rightarrow (\text{Violated}(e_4, c) \leftrightarrow \neg\phi) \tag{5}$$

This says that the obligation of SkyHi Builders to pay Steelman's Warehouse is violated, in terms of clause c, if, and only if, SkyHi Builders did not actually pay £25,000 to Steelman's Warehouse. Kimbrough here uses a variant of the Anderson's [And58] reduction, which says that 'ϕ ought to be the case' unpacks to 'necessarily, if ϕ is not the case, violation happens'.

7 An Implementation of Kimbrough's Disquotation Theory

Kimbrough's logic is, by and large, appealing; certain alterations were proposed above. What we desire is a database representation that mimics this logic. We suggest an implementation in Java which models the quoted contents $[\phi]$ as a query - specifically, a query that returns occurrences. The quoted contents cannot be modeled as facts as this would imply that the occurrences actually occurred, when in fact they were only *described*. That is, the occurrences enclosed in a disquotation operator should not be regarded as actual occurrences, but *descriptions* of occurrences. For instance, if obliged occurrences were regarded as actual (rather than described) this would imply that all obligations are kept, when indeed obligations may be violated should the described occurrences never happen. Thus it is proper to model only the criteria that describe the occurrences, since these describe what nature of occurrence was obliged, without the undesirable implication that the described occurrences were actually realized. We provide a *continuous query mechanism*,[8] which determines incrementally, as data is added to the database, which stored queries produce new results or consume existing results, and which stored queries begin to, or cease to, overlap. Such a mechanism enables the system to determine whether an occurrence is obliged, permitted, or prohibited, by determining whether the occurrence fits a stored query.

When we store obligations, permissions, prohibitions, and powers, what we are putting into the database are representations of expressions with propositional content: e.g. x permits that P, it is obligatory that Q, y prohibits that R, S brings about T. We know that the *expression* is true since we can be sure the obligation, permission, prohibition, or power was *once given*, that is *expressed* (even if it has since been overridden). However, we do not assume that the *disquoted propositional content* - P, Q, R, S, or T - is true, since permissions are not always used, obligations are not always met, prohibitions are not always violated, and powers are not always exercised. We must then use a query - a descriptive set of criteria - to store the disquoted propositional content, rather than storing the disquoted propositional content as an actual occurrence. This captures the intuition that the disquoted contents was *described*, but may not have actually *occurred*. The disquoted contents is not an *actual* occurrence, but a *description* of an occurrence or set of occurrences. The benefit of using stored queries - that is, stored descriptions - is that, when and if the content (occurrence or set of occurrences) does eventuate, we can detect that the occurrences (eventualities) are covered by a stored query. A continuous query mechanism, or 'coverage checker', is employed for this purpose. Upon insertion of a new occurrence, the coverage checker determines which stored queries covers the new occur-

[8] See §10.4, and [AEB02b], [AEB02a], and [AEB02c].

rence, and which expressions of permission, obligation, prohibition, or power - and hence, which clauses of which contracts - pertain to the occurrence.

This section revisits examples of assertions, prohibitions, permissions, and obligations above, demonstrating implementations of Kimbrough's constructs as well as additional notions we have found necessary.

7.1 Assertions

The case of assertions is unique, in that we have decided not to adopt the disquotation operator for reasons of the inherent subjectivism of our approach. By this we mean that we have chosen a world-view where all recorded facts are taken as subjective opinions. For our purposes, we do not find it meaningful to determine whether an assertion is 'veridical' since we are unable to compare to an objective truth: only subjective opinions - X is the case according to this clause, and Y is the case according to another clause - are stored.

For the anomolous case of assertions then, our approach has been to supplant the disquotation operator with an alternative: we append the conjunct IsAccordingTo(e, ClauseX) to all occurrence descriptions. This captures the notion that all occurrence descriptions are prima-facie construals relative to an utterance (identified by a system-defined clause identifier). Our database representation assumes that, while objective truth exists, it may sometimes be arguable as to what it is - all occurrence descriptions are relative to some clause. For instance, 'The clerk, John, bought steel, *according to Clause 8*' does not mean that the steel was certainly bought (for other regulations may override); only that the Clause 8 certainly provides such. That 'John bought steel' is therefore the opinion of Clause 8, and is merely prima facie, not conclusive, evidence of buying. Our rules must be specific. We must differentiate 'buying according to Clause 8' from 'buying according to Clause P.1', since it is possible, in the case where clerks are not empowered to buy steel and John is a clerk, for there to be a buying in terms of the former clause, but not in terms of the latter. In contrast to many traditional views of data representation, in our database we record what has been *said*, rather than merely what *is*: when we ask the database what *is*, we are more specifically asking what *is according to a certain utterance (clause)*. Objective truth there is, but we may never know *what* it is: comparing and contrasting assertions may guide us towards it.

It is worthwhile to note here that many approaches to evaluating subjective evidence are available. In our *qualitative* approach, occurrences according to a particular clause are brought about, through a function, whenever there are occurrences in a particular domain - that is, whenever an occurrence fitting a convention is recognized. An interesting and complementary *quantitative* approach to assessing the believability of subjective opinions of various agents in a distributed setting is provided in Dimitrakos and Bicarregui [DB01] and Daskalopulu, Dimitrakos, and Maibaum [DDM01]. In that

work, a believability metric is associated with propositions, where such metric is computed by a numeric combination of weighted opinions. Roughly, Dimitrakos and co-authors have it that 'pizza A is delivered' if Susan believes with 0.8 probability that it was delivered, Peter believes with 0.2 probability that it was delivered, and Susan's opinion is weighted more strongly than Peter's, based on their reputations and credibility in the area. In contrast, we might have it that 'pizza A is delivered, according to Clause 8.6 of the delivery contract between Peter and Susan' whenever Peter says it has been delivered, irrespective of Peter's credibility. Such a construal of what 'delivery' means is common in 'customer satisfaction guaranteed' contracts. Clause 9.1 of the Uniform Commercial Code may specify different criteria as to what it means for a good to have been delivered, and we may need to choose, in accordance with explicit legal principles, which construal overrides in the circumstance. Our approach to subjectivity of information is one of 'locality of inference', whereas Dimitrakos and co-authors deal with the aspect of 'reliability of evidence'.

7.2 Prohibitions

We distinguish here between two types of prohibitions: violable prohibitions, and inviolable prohibitions (legal disabilities).

Violable prohibitions: Violable prohibitions admit the possibility of violation. This is the sense of prohibition used by Kimbrough. If an occurrence fits the description of occurrences prohibited by a particular clause, it can be said to be prohibited in terms of that clause, and its existence brings about a violation of that clause. Consider Clause P.3 ('payments of more than £10,000 to suppliers are prohibited') from our application scenario of 2. The 'violable prohibition' sense of this clause can be modeled as shown in Table 2. Here, Query10 is a pointer to a query describing the set of prohibited occurrences: in our example, occurrences of a supplier being paid more than £10,000. §10.3 describes how such a query would be represented and stored. violating_function1 is an identified function, whose domain is Query10 and whose range is a set of occurrences violating_function1(Query10). We append the occurrence type produced by the function to the start of the function name to make the output range more clear and more easily accessible to the query mechanism. Therefore, an occurrence, paying1, as defined in Table 1, which is covered by Query10 and hence in the domain of violating_function1, would produce an occurrence violating_function1(paying1), which can be seen (Table 2) to be an occurrence of violating, where it is prohibiting1 that is violated. The function construct is comparable in some senses to Jones and Sergot's [JS96] counts-as connective, $\Rightarrow_S$, and Goldman's conventional-generation connective, though our functions are relativized to an identified clause, C, rather than relativized to an institution, S, as for the counts-as connective. Thus, it is the clause that deems that a certain occurrence is brought about, rather

Table 2. A schema for storing a violable prohibition

Occurrence	Role	Participant
prohibiting1	prohibited	Query10 = occurrences where more than £10,000 is in role=paid ∩ occurrences where a supplier is in role=payee
	isAccordingTo	Clause 2.4
violating_function1	domain	Query10
	violated	prohibiting1
	isAccordingTo	Clause 82

Table 3. A schema for storing a disability or immunity (inviolable prohibition)

Occurrence	Role	Participant
counting1	isAccordingTo	Clause P.1
	counted	Query200 = occurrences where a clerk is buyer ∩ occurrences where steel is bought
	count	0

than an institution. This is necessary because clauses are atomic, whereas current institutional interpretations can only be selected once all *prima facie* atomic clausal interpretations have been compared.

Again, we point out that all provisions in a contract should be regarded as subjective construals. 'The prohibition is violated' should not be taken to mean that the prohibition is *authoritatively* violated; but should rather be taken as a partial rendering of the sentence 'The prohibition is violated *according to some clause*'. Another clause may provide that the prohibition is not violated, and a choice as to which clause to accept as authoritative is then required. An institution states multiple clauses, and the contents of clauses may conflict. Determining the overarching construal of the institution involves choosing which clause applies in the circumstance.

Inviolable prohibitions (legal disability or immunity): Inviolable prohibitions cannot be violated, since the prohibited party simply doesn't have the legal power to bring about the prohibited state. Under Hohfeld's [Hoh78] terminology this would be called *disability*. We could take Clause P.1 of our application scenario (§3) as specifying that clerks have no authority to purchase steel. Table 3 illustrates the representation of this sense of the clause: it specifies that the number of purchases of steel by a clerk under this clause is zero (the count of results produced by Query200 is 0). This means that no action by them can bring about a purchase in terms of this clause.

A similar construction can be used to implement Hohfeld's [Hoh78] notion of *immunity*: under immunity, no action by any party can bring about a certain state of affairs (e.g., no party can bring about an occurrence of a particular party being obliged to do something). The party that benefits from this immunity is said to be immune. In Table 3, the description (Query200)

does not specify the seller; the implication is that steel purchases by a clerk from any seller may not come about. This means that, according to this clause, everyone has immunity from entering into steel purchases with a clerk.

The relationship between violable and inviolable prohibitions, and soft and hard constraints in databases is oblique. A soft constraint in a database is one where the update is allowed, but a violation is flagged. The violation may or may not be persistent; that is, an exception may be generated, but typically the exception is logged in a text log, that cannot be easily queried. A hard constraint is one where the update is rejected, but the attempt to update may be logged, again typically in a text log that cannot be easily queried. A violable prohibition is somewhat similar to a soft constraint. However, in the case of violable prohibitions, the violations are persistent: their existence may be ascertained by querying the database. The database may, of course, also notify an interested party of the violation. An inviolable prohibition is subtly different from a hard constraint in the database: updates are not rejected, but simply have no harmful effect. A purchase by John, who is not empowered to make purchases, is still recorded as a purchase (since it is a purchase in some sense), but cannot be a 'purchase according to Clause P.1 of company policy'. Since only 'purchases according to company policy' bring about obligations, John's purchase does not bring about any obligation to pay, and consequently has no effect.

Finally, note that when a policy states 'clerks may not purchase steel', this may be intended as implying the existence of *either* a violable prohibition, an inviolable prohibition, or *both*. We do not intend the choice to be an either/or choice between the types of prohibition, since often the intention is that there is both a violable and an inviolable prohibition in existence. English specifications of policy are problematic since it is not clear whether 'may not' is intended to imply the existence of a violable prohibition, *or* an inviolable prohibition (legal disability or immunity), or *both*. Our purpose in distinguishing between violable and inviolable prohibitions is so that we may disambiguate English specifications of policy and be more particular as to whether we mean for there to be a violable prohibition, an inviolable prohibition, or *both*. In this case, assume we mean for there to be both a violable prohibition and a inviolable prohibition (legal disability or immunity). If the clerk indeed proceeds to make a purchase, he violates the violable prohibition *and* his purchase is still covered by the inviolable prohibition which prevents it from having an effect. It is tempting to say that the clerk violated the prohibition by performing an act which he is not empowered to perform, but this is a little misleading. In fact, the clerk did not exercise his power in this case, since he performed a purchase, but not a 'purchase in terms of Clause P.1' so no legal power was really exercised in that respect (because the clerk in fact had a legal disability). Yet he is still guilty of a violation, and perhaps subject to sanctions against him, since it was purchases of any sort that were forbidden, and even though he did not perform a 'purchase in terms of

Clause P.1', he did perform a purchase in some other sense. The fact that the purchase is not effectual in terms of Clause P.1 does not mean that it doesn't violate a prohibition against all purchases, since it is still nonetheless a purchase in some other sense.

7.3 Permissions

'Employees older than 25 may buy steel (Clause P.2)', from our application scenario, is also an ambiguous expression. It may refer to employees older than 25 being *permitted* (allowed) to buy steel, or to employees older than 25 being *empowered* (that is, legally capable in terms of a clause) to buy. The former is dealt with here, whilst the latter we leave for the next subsection (§7.4).

We distinguish between two types of permissions: violable and inviolable permissions.

Violable Permissions : These are the Benthamite 'vested liberties' referred to by various authors [Lin77,Mak86]. This sense of permission can be represented as a violable prohibition (§7.2), where the prohibited occurrences are described as 'any attempts to prevent the permitted occurrences'.

Inviolable Permissions : This is the sense of permission used by Kimbrough. Inviolable permissions are not directly comparable to Benthamite 'naked liberties' since the concept of naked liberty entails only that interference by other parties is not prohibited, but does not explicitly specify that performing the permitted action can never lead to a violation as Kimbrough [Kim01] and Lee [Lee88a], following Anderson [And58], suggest. That is, by 'inviolable permission' we specifically mean to capture the idea that performing any action covered by the permission does not lead to any violation. This subtlety is not captured by the term 'naked liberty'. Consider an example. '*Employees older than 25* are permitted *to buy steel*' embeds the query 'select purchases by employees older than 25 of steel'. That is, according to this clause, any occurrence fitting this query is permitted and does not bring about any violation. The permission implies the count of violations brought about by occurrences of this nature is zero. Table 4 illustrates. Inviolable permissions are comparable in some sense to Hohfeldian *privileges*; privileges [Hoh78] imply no duty to refrain exists, i.e., no violation is brought about by indulging in the action.

As we pointed out for prohibitions above, we do not mean for the terms 'violable' and 'inviolable' to imply that the two types of permissions cannot co-exist in relation to a particular set of described occurrences. Nor do we mean that a particular English sentence implies one or the other. Indeed, violable and inviolable permissions often do co-exist, and a given English sentence may ambiguously imply, *either* or *both*. Consider "Employees older

Table 4. A schema for storing an inviolable permission (privilege)

Occurrence	**Role**	**Participant**
permitting1	permitted	Query300 = occurrences where employee older than 25 is buyer (occurrences where steel is bought
	isAccordingTo	Clause P.2
counting1	counted	Query400 = occurrences of violating brought about by occurrences in Query3
	count	0
	isAccordingTo	Clause 2.6

Table 5. A schema for storing a power

Occurrence	**Role**	**Participant**
buying_function1	domain	Query500 = (occurrences where an employee older than 25 is buyer ∩ occurrences where steel is bought) - occurrences where Clause P.2 is isAccordingTo[9]
	isAccordingTo	Clause P.2
	buyer	participants in role buyer in occurrence |Query500|[10]
	bought	participants in role bought in occurrence |Query500|

than 25 may buy steel (Clause P.2)". This may be taken to imply *either* or *both* of:

- Anyone who interferes with such an employee's right to buy steel is in violation (a violable permission)
- Such an employee buying steel does not bring about a violation in terms of this clause (an inviolable permission).

Should another clause prohibit an individual who is such an employee from buying steel, we can see that this prohibition conflicts with the permission, since the prohibition implies that violations are brought about by such purchases, whereas the permission implies that no (zero) such violations are brought about.

7.4 Powers and Liabilities

In our variation of Jones and Sergot's usage, powers encode the ability of an actor to bring about a state-of-affairs according to a clause. Consider the rule 'Clause P.2: Employees older than 25 may buy steel'. The reading of this as a power of the employee may be represented as shown in Table 5.

In Table 5 the function buying_function1 encodes the power to bring about a purchase. Clearly, the only means of bringing about a *purchase in terms of Clause P.2* is to insert an occurrence in the domain (Query500) of the function. Assume buying4 is a purchase of steel by the 40-year old manager Marge. buying4 is covered by Query500 and therefore is in the domain of the function buying_function1(). Substituting buying4 for Query500 in buying_function1(Query500), the function therefore produces the occurrence instance identified as buying_function1(buying1). However, it should be noticed that, as the database wrapper automatically tags each occurrence with a clause identifier upon addition to the database, simply adding an occurrence of buying to the database, does not make it a *purchase in terms of Clause P.2*. So a purchase, say buying6, of gold by a dispatch clerk would not be construed as a *purchase in terms of Clause P.2*, since buying6 is not covered by Query500 and therefore not in the domain of buying_function1.

Notice that the mechanism of using functions to bring about states-of-affairs-according-to-a-clause implements not just *powers*, but also *liabilities*, which, according to Hohfeld [Hoh78], are the correlative of powers. Hohfeld does not here use 'liability' in the accounting sense of debt or "having an obligation towards another party"; rather, he makes use of 'liability' in the sense of "being subject to having one's legal relationships altered by another party". As Hohfeld explains, his sense of liability does not necessarily imply disadvantage to the liable party; liability implies merely that the party is subject to having their legal relations altered by virtue of an occurrence. The notion of power implies that an actor is empowered to bring about a state of affairs[11] through their own action; what Hohfeld terms 'volitional control'. The notion of liability is similar, though it implies that the state

[9] This third set criterion is necessary in order to prevent infinite recursion: that is, to prevent purchases according to Clause P.2 from bringing about purchases according to Clause P.2. The symbol '-' can be read '... but not ...'.

[10] We use the notation |...| to refer to the specific item in the domain of the function that caused the function to produce a value (i.e., the variable bound to the function instantiation). This gives us a means of referring to the originating occurrence that brought about the state of affairs produced by the function. We can refer to the attributes of said originating occurrence, by embedding the reference |...| into query expressions from our language (such as participants in ...) as shown. To give an example: Assuming the occurrence buying1 is in the domain denoted by Query500, it then brings about an occurrence buying_function1(buying1) as shown in Table 5. Substituting buying1 for |Query500| in the query expression `participants in role buyer in occurrence` |Query500| we obtain `participants in role buyer in occurrence` buying1. The result of this query is the participant in the buyer role in the occurrence buying_function1(buying1).

[11] Often, a deontic state-of-affairs such as being-obliged in terms of a particular clause may be brought about. However, the state of affairs need not necessarily involve obligations, prohibitions, or permissions. It may be, for instance, buying or owning in terms of a particular clause. The state of affairs brought about,

of affairs may be brought about by another party, by an environmental occurrence (such as the passing of a certain date), or even, as we argue, by an action of the liable party. We contend that power and liability are in fact best treated using the single function device to describe how occurrences fitting a convention bring about other occurrences. Consider that an occurrence of violating a prohibition against interference may, through a function being_obliged_function1(...), bring about an occurrence of being-obliged (a so-called 'secondary' obligation) to pay damages. For Hohfeld [Hoh78], this situation is awkward since X is both the liable (to have his legal relations changed) and empowered (to change legal relations) party. Hohfeld says a person holding a 'power' has the legal ability by doing certain acts to alter legal relations; the one whose legal relations will be altered if the power is exercised is under a 'liability'. And yet here, X clearly has a power to alter legal relations (by interfering and violating the prohibition), and a liability to have his legal relations altered (by interfering). This situation is awkward because Hohfeld sees his legal relations as being 'relations of one individual with another', yet here X has a relationship with himself. As is evident from our function device, we do not see it as problematic that any occurrence, whether intentional or not, positive or negative, can bring about legal relations upon any person. The liable party is often a passive participant in an occurrence in the domain of the function which encodes the liability, but he may just as well be an active participant in said occurrence (as in the case of interfering) or not participate in the occurrence at all (as in the case where an obligation is, say, contingent on the occurrence of an environmental event).

7.5 Obligations

In our application scenario, "SkyHi is obliged *to pay Steelman's £25,000 by 1 September 2000* (Clause C.1)" embeds the query: "select the first payment, before 1 September 2000, of £25,000 by SkyHi to Steelman's" since that is the nature of the obliged occurrence. Only occurrences of that nature can fulfill the obligation. It can be noticed that the *duty-bound* party is the agent specified in the occurrence description (in this case, SkyHi). The obliged occurrence may not exist yet, but when it does, we can say, by its nature, that it was obliged if it fits the query. If in the future the query is 'filled' - that is, the count of items fitting the query is the maximum number of results the query can produce; in our example, one result - the obligation can be said to be 'fulfilled'. In the absence of occurrences fitting the description - that is, when the count of items fitting the query is zero - the obligation is not fulfilled. If the count of items fitting the query is less than one after

though not necessarily deontic, is essentially a *legal* state of affairs: i.e., one recognised as such by some clause of a normative system.

Table 6. A schema for storing an obligation, its fulfillment and violation conditions, and general liability to damages

Commentary	Occurrence	Role	Participant
SkyHi *being-obliged* (according to Clause C.1) to pay, before 1 September 2001, to Steelman's, £25,000 for shipment.	being_obliged1	according-to	Clause C.1
		obliged	Query19 = first occurrence of SkyHi paying Steelman's £25,000 for the shipment, where paying is before 1 September 2001
Filling of the above-mentioned query *bringing-about*, in terms of Clause L.1, fulfillment of the obligation (an occurrence of *fulfilling* the obligation).	fulfilling_function1	domain	Query700 = first occurrence of count (Query19) being 1
		fulfilled	being_obliged1
		IsAccordingTo	Clause L.1
Failure to fill the above-mentioned query after the deadline *bringing-about*, in terms of Clause L.2, violation of the obligation (an occurrence of *violating* the obligation).	violating_function1	domain	Query800 = first occurrences of count (Query19), after the deadline (1 Sept 2001) being less than 1
		violated	being_obliged1
		IsAccordingTo	Clause L.2
Successful instigation of prescribed procedure following violation of obligations, *bringing-about* , in terms of Clause L.3, secondary obligations to pay damages	being_obliged_function2	domain	Query900 = occurrences of successfully instigating prescribed procedure following occurrences of violation of occurrences of being-obliged
		IsAccordingTo	Clause L.3
		obliged	Query1000 (secondary obligation) = first occurrence of paying damages for the violation for which legal action was instigated in |Query9|

the deadline (1 September 2000), the obligation is violated[12]. The violation usually brings about secondary obligations upon the violator; in this case Clause L.3 of our application scenario tell us there is a secondary obligation to pay damages for violation. Often the violator is not *personally* liable for the violation - for example, in the case of insurance or other transfer of liability - as the secondary obligations may fall upon other parties. So, the descriptions of occurrences obliged by the secondary obligation may be of occurrences where the agent (i.e., duty-bound party) is the original violator, or where the agent is some other party.

Our obligation of SkyHi to pay Steelman's (Clause C.1) may be represented as shown in the Table 6, where Query19 is a pointer to a stored query that describes the set of obliged occurrences in the primary obligation, being_obliged1). Representation of obligation fulfillment (Clause L.1), violation (Clause L.2), and liability to damages (Clause L.3), is also shown.

With reference to Table 6, we are able to specify the following parties mentioned as important in §6:

- party responsible to act: payer mentioned in Query19 describing the primary obligation (SkyHi Builders)
- party responsible in surety: payer mentioned in Query1000 describing the secondary obligation (happens to be SkyHi again)
- party directly benefiting from the obliged action: payee mentioned in Query19 describing the occurrences obliged under the primary obligation (Steelman's Warehouse)

[12] Note that if the count, before the deadline, of items fitting the query is zero, the obligation is not violated, but rather 'not (yet) fulfilled' (which is not to imply that it ever will be fulfilled).

- party empowered to initiate recourse: instigator mentioned in Query900 describing the occurrence that brings about the secondary obligation (happens to be Steelman's again)
- party entitled to outcome of recourse: payee mentioned in Query1000 describing the secondary obligation (Steelman's, again)
- party issuing the norm: the party uttering, or organization or document associated with, the clause (for example, the Clause C.1 is associated with the contract between SkyHi and Steelman's)

As each obligation is *according to a particular clause*, these obligations may be considered as a representation of the notion of *prima facie* obligations discussed in the deontic logic literature [PS97]. We ought to note at this point, to avoid confusion, the distinction between *primary* obligations and *prima facie* obligations put forth in Prakken and Sergot [PS97]. There is one contrast between *primary* and secondary obligations (the latter coming into being upon violation of the former), and another contrast between *prima facie* and all-things-considered obligations (the former being on-the-surface, if clauses are considered in isolation, and the latter being what is said by the contract, act, or law as a whole). Our treatment of obligations as being 'according to a particular clause' and our selection of a particular clause during conflict resolution is intended to deal with the distinction between *prima facie* and all-things-considered obligations. The distinction between *primary* and secondary obligations, we have seen (Table 6), is dealt with through occurrences of violating and functions that bring about secondary obligations as a result of those occurrences of violating.

A particular English sentence (clause) may define a single obligation, such as the single occurrence of *being-obliged (by that clause)* to pay Steelman's Warehouse £25,000 by 1 September 2000. However, a single clause may also define how multiple separate obligations are brought about. Consider Clause L.3 of our application example, which effectively says that 'each occurrence of successfully instigating the prescribed procedure after each violation of an obligation brings about a separate obligation to pay damages'. Interpreting Clause L.3 as implying the existence of a single occurrence of being obliged would be incorrect, as this would implies a *joint* or *collective* obligation by all liable parties together to pay damages to all aggrieved parties together. Rather what is intended is that there are *several*, *separate* obligations. The several obligations may be formally defined as:

f : occurrences of successfully instigating $\rightarrow$ occurrences of being-obliged

This reads that a function f maps occurrences of successfully instigating to occurrences of being obliged. For example, in Table 6 f is the function being_obliged_function2. That is, the identified function, being_obliged_function2, takes as its domain occurrences of successfully instigating and produces occurrences of being-obliged. In Table 6 we have defined the domain of the function using a query, Query900, and specified the participants in the occurrences in

the range of the function. Bindings from a participant in an output occurrence to a participant in the corresponding input occurrence that produced that output may be represented using |Query900| embedded in any valid query expression. |Query900| in the query is then substituted with the corresponding value from the domain of the function. For example, assuming instigating1 is a successful instigation, first occurrence of paying damages for the violation for which legal action was instigated in |Query9|, becomes first occurrence of paying damages for the violation for which legal action was instigated in |instigating1|, for the occurrence being_obliged_function2(instigating1) brought about by the occurrence instigating1. It should be clear from this discussion that, in the case of clauses that generate multiple obligations, individual obligations can be identified by their function identifier (e.g. being_obliged_function2) and the identifier of the occurrence that generated the obligation. For example, as we saw above, instigating1 generated the particular obligation being_obliged_function2(instigating1) where being_obliged_function2 was a function defined in Clause L.3. For ease of lookup, we always prefix the function identifier with the occurrence type of the occurrences it produces: therefore instead of simply function2(...) we store being_obliged_function2(...) so that we can see that the function produces occurrences of being_obliged.

Functions may also be used to represent the generation of individual, separable obligations on each member of a group. However, equally important, and frequently mentioned in the deontic logic literature, is the ability to represent *collective* obligations, where no specific individual bears the personal responsibility. Moffett and Sloman [MS93] mention the notion of collective responsibility, but no implementation is provided in the subsequent work on Ponder [DDLS01]. Ponder distributes an obligation to *all* subjects of the obligation—that is, distributes multiple, separate obligations—whereas a collective obligation is meant to capture the idea that the obligation is one obligation and *any* of the members of the group could fulfil the obligation. Only a single occurrence, by any one of the group members responsible, needs to be brought about to fulfil the collective obligation. Consider the obligation, say being_obliged4, of any report-writing component to generate year end reports on 30 September 2001. Assume there are many such components capable of fulfilling the obligation. In our approach, the obliged occurrence can be described using the query "the first occurrence of generating, on 30 September 2001, year end reports, by a report-writing component". Clearly, only one suitable occurrence of report-generating is required to fill this query and therefore fulfil the obligation, and any of the report-generating components, having seen the obligation, can bring about an occurrence that fills the query. Though we do not deal formally with such matters here, some coordination mechanism is obviously desirable to ensure that no component tries to fulfil the obligation when another component is busy with it, and looks likely to succeed. Certainly, there is some indeterminacy here, as any of a number of components could fulfil the obligation, but this is not problematic since we

require only that the obligation be fulfilled by some member of a group and do not prescribe who specifically must fulfil it. Our high level requirement need not be polluted by specific indication of which particular component is responsible, but can nevertheless meaningfully specify responsibility of a group. Our representation is able to assess when the group's responsibility has been fulfilled. We of course assume that all components are diligent , i.e., that one of them will take up, or be assigned, the task. And in the event that none do, we are equally able to assess that the group's obligation has been violated, since no occurrences fitting the query described in the obligation have occurred by the deadline.

8 Contract Provision Monitoring

By contract monitoring we mean that, when workflow occurrences are added to the database, our implementation sees to it that the necessary contractual consequences are inferred. This usually includes the inference of obligations that effectively drive the system by identifying what it is that capable components ought to perform next. Coordination and mutual understanding are facilitated as components are able to use the database to determine which legally-recognized states of affairs hold: for instance, they may unambiguously determine whether a purchase is valid in terms of a particular provision stored in the database.

Multiple modes of contract provision monitoring are possible. In cases where third parties bring about occurrences, violation may not be avoidable, and *immediate detection* of such violations is required. Often, however, policies are not sufficiently critical to warrant constant monitoring, and *delayed detection* may be suitable.

8.1 Immediate Detection

Often, detection of a violation is required immediately after an occurrence is added to the database. This section describes how immediate detection may occur, and explains the basics of the coverage-checking mechanism employed. The coverage-checking mechanism makes use of covering relations between queries, and partial re-evaluation of dirtied queries. The intention is to determine when the results of stored queries change as a new data is added. For instance, if a new occurrence is covered by a stored query that describes the occurrences in the prohibited role, in a prohibition, this indicates that the occurrence is prohibited under that prohibition.

A taste of the coverage-checking algorithm is best given through an example. Take the policy 'payments of more than £10,000 to suppliers are prohibited'. As illustrated in §7.2, this is represented by storing an occurrence of prohibiting, prohibiting1, with a reference to Query10 in the prohibited role of that occurrence, where Query10 describes the occurrences that are prohibited.

The storage of queries, and determination of covering relationships between queries, is described in §10.3 and §10.4. Now, assume that this prohibition, and its associated query (which describes the prohibited occurrences) are stored in an empty database. Then, assume that we record, in this database, that Steelman's Warehouse is a supplier of SkyHi Builders, by inserting an occurrence, being_supplier1, as described in Table 1, into the data-store. Upon insertion of the rows for being_supplier1 the coverage-checking algorithm kicks in. The coverage-checking algorithm determines that the new occurrence, being_supplier1, is not prohibited, since no relevant queries cover it. Now assume that the payment, paying1, is inserted. The coverage-checking algorithm determines that paying1 is covered by Query10. A quick lookup shows us that Query10 is in the Participant column in a row in the database where prohibiting1 is in the Occurrence column. As paying1 is covered by Query10, which describes the set of prohibited occurrences, paying1 is therefore prohibited.

Our incremental coverage-checking mechanism determines which queries cover an item newly added to the database. The incremental nature of the algorithm is important as tens of thousands of occurrences may be stored in the database. Re-executing every query stored in the database each time an occurrence is added is infeasible, especially as most results will be unchanged. It is important therefore to only re-execute queries whose results may have been changed. This may involve re-expressing computation-intensive general queries as more efficient specific queries: a process we call 'partial re-evaluation'. For large data volumes, a query execution approach is likely to be more efficient than a vanilla theorem-proving or logic programming approach, as the query execution approach incorporates query optimizers which take into account data profiles (predicate selectivity) when executing a query, whereas basic theorem provers and logic programs typically do not concern themselves with such execution efficiency issues.

The computationally-intensive immediate conflict detection algorithm, which checks which queries cover every incoming occurrence, is sometimes infeasible for large data volumes. Another alternative, discussed in the next section, is to buffer occurrences and perform delayed detection.

8.2 Delayed Detection

The previous section described how the coverage-checking mechanism may be employed for immediate detection of violation: as each occurrence is added, it is checked against stored policies. However, some policies do not require such rigid enforcement. Stationing a full time security guard at the office supplies cupboard would likely be a sub-optimal allocation of resources. Similarly, checking all occurrences against policies is inappropriate for those occurrences that, individually, are of little consequence. Delays in detecting violation may be traded off against the computational cost of detecting violation immediately. Periodic or aperiodic detection, such as 'at the end of the

day', 'on every second Tuesday of the month', or 'at off-peak times of system utilization', may be all that is called for. Thus, to reduce the monitoring burden, configurable evaluation based on the time-criticality of violation detection should be possible, to allow non-critical policies to be assessed less frequently.

Two primary mechanisms for delayed detection are possible:

1. Top-down query re-evaluation: periodically (or aperiodically) executing the queries associated with selected policies. This would be inefficient if many of the queries were unchanged during the interval.
2. Bottom-up batched occurrence detection: Batch checks of unique identifiers across multiple occurrences may reduce computational cost. In batch checks, unique identifiers added to the database within an interval are fed into the coverage-checking mechanism after a batch of occurrences, rather than after a single occurrence. The computational saving achievable by this mechanism depends on the frequency of updates of each particular concept identifier. For concept identifiers frequently added to the database the saving would be greater as the concept identifier would be fed into the continuous query mechanism only once for all updates during the interval. For instance, assuming the addition of a thousand occurrences of paying, each with roles payer, and payee. An immediate detection mechanism would check which queries mention the role payer one thousand times, and which mention the role payee one thousand times. A batch mechanism would check once which queries mention payer and would check once which queries mention payee. Adelberg, Garcia-Molina and Widom [AGMW97] illustrate, in a stock market application, that a mechanism for batching updates and computing the net effect at the end of a delay window is efficient when data is input in short bursts of similar data.

The need for more delicate control of when inferencing occurs is another reason for our choice of Java. Delaying inferencing to prevent bottlenecks during intervals when the data update rate is high is not provided in logic programming and theorem proving approaches, which would need to be supplemented by data-preprocessors, such as the delayed detection mechanism mentioned here.

9 Contract Performance and Enforcement

Performance and enforcement of a contract may be roughly divided into: intervention, prevention by refusal, and prevention by construal. We treat performance and enforcement together since we view enforcement as overlapping with performance: part of enforcement is the performance, typically by a supervisory component, of obligations that arise from violations of a contract. Performance is via *intervention*: we take it that components are

diligent, though perhaps not always successful, in attempting to perform their obligations. We assume that components consult the central active contract database to determine their current obligations and attempt to fulfill them. Enforcement may, however, involve not just intervention, but also prevention. In the case of policies where the consequences of violation are high and control over activity is possible, *prevention by refusal* is appropriate. A refusal of a request to execute contemplated action avoids violation. Where control over third-party activity is not possible, *prevention by construal* can be employed. Here, any occurrence not fitting the 'rules of the game' is not deemed to be an occurrence of a certain type in terms of 'the game' (the law), thereby preventing it from having consequence and effect.

9.1 Intervention

As we noted earlier (§4) our architecture presents a central database where contractual provisions, workflow occurrences, and consequent contractual construals derived from those workflow occurrences are stored. The database stores both the application's specification—that is, the provisions which specify what can, should, and should not happen in various circumstances—and the operational workflow occurrences. This dynamically changing contract database is consulted by the various implementation components to determine what states of affairs hold in terms of the contract, and what occurrences to perform, or refrain from performing, next based on prevailing obligations, permissions, and prohibitions. Ponder [DDLS01] provides a policy deployment mechanism whereby language-specific imperative scripts or access control policies are deployed to distributed components. We provide no such mechanism, which in any event limits enforceability of the policies to only the specific platforms supported by the script generators. Instead, we rely on the heterogeneous components to consult the database and determine their obligations, by looking up occurrences of being-obliged which pertain to them. They can then attempt to fulfill these obligations. Alternatively, we allow facilitator components (e.g., a fulfillment scheduler) to do the consultation and invoke the implementation components. This process by which diligent components determine and fulfill relevant obligations is called *intervention*, and is one contract provision enforcement mechanism. Prevention, in various modes (by *refusal* and by *construal*), is another possible enforcement mechanism.

9.2 Prevention by Refusal

In the case of policies where the consequences of violation are high and control over activity is possible, *prevention by refusal* is appropriate. Here, a decision not to execute contemplated action avoids violation. Prevention via refusal requires analytic detection of a conflict between a contemplated set of occurrences and a prohibition.

Again, this is best illustrated through an example. Let us record, in a new and otherwise empty data store, that Steelman's is a supplier of SkyHi, by inserting an occurrence, being_supplier1, as described in Table 1. Assume that there is no occurrence of paying yet. Further, assume that a prohibition against paying more than £10,000 to suppliers is provided by Clause P.3. As illustrated in §7.2, this is represented by storing an occurrence of prohibiting, prohibiting1, with a reference to Query10 in the prohibited role of that occurrence, where Query10 is the stored query:

```
Query10 = occurrences where [>10,000] is [=paid-amount] ∩
          occurrences where [participants in the role [=supplier]
                             in [occurrences of [being_supplier]]]
                             are [=payee]
```

(Query describing *prohibited* occurrences)

Consider now that SkyHi Builders is obliged, according to Clause C.1, to pay, before 1 September 2001, to Steelman's Warehouse, £25,000 for a shipment. Assume that this payment has been contemplated, but not effected, i.e., it has not yet occurred; no occurrence of paying has been added to the data store. As shown in §7.5 the obligation is stored by storing an occurrence, being_obliged1, where the *obliged* (that is, *contemplated*) occurrences are described by the query Query19:

```
Query19 = first occurrence of SkyHi paying Steelman's £25,000
        for the shipment where paying is before 1 September 2001
```

or, more formally:

```
Query19 = 1st of [
          occurrences where [=SkyHi] is [=payer]
          occurrences where [=Steelman's] is [=payee] ∩
          occurrences where [=25,000] is [=paid_amount] ∩
          [participants in role [=allocated]
                        in [occurrences where [=shipping1]
                                              is [=allocatedTo]
                                              and [=FIFO]
                                              is [=allocationBasis]]] ∩
          [participants in role [=occurred]
                        in [occurrences where [<1 September 2001]
                        is [=occurredOn]]]]
        in [ascending] [temporal] order
```

(Query describing *obliged* occurrences)

Now, comparing the description of the *obliged* occurrences (Query19) to other queries stored in the database, even in the absence of data, our coverage-checker (§10.4) finds that Query19 analytically overlaps with Query10. Further, Query10 is a description of *prohibited* occurrences (by virtue of Query10's

participation in prohibiting1). We have thus shown that what is obliged (the description of the obliged occurrences) in this context is covered by what is prohibited (the description of the prohibited occurrences). This indicates a *conflict*. Our analytic, specification-time query overlap determination mechanism can be compared to the empirical approach employed in Damianou et al [D$^+$01] which requires that actual run-time data exist in order to be able to determine that an item is covered by two queries (or what they call 'domain scope expressions'). Their approach is therefore not able to detect certain conflicts until run-time.

If we assume the existence of a choice principle that says the prohibition overrides the obligation in this case, our prevention by refusal mechanism tells us that no operation that might bring about an occurrence of paying to fulfill the obligation should be allowed to execute. That is, if a component is contemplating executing an operation that would bring about such an occurrence, and it requests permission to go ahead from some supervisory or control component, the permission should be refused. We leave it to the implementation component (e.g., firewall, gateway, or access control layer) to see to it that the refusal is enforced. The intention of our 'prevention by refusal' mechanism is merely to illustrate that enforcement decisions can be made by determining whether the contemplated (obliged) action is covered by a prohibition.

9.3 Prevention by Construal

It may be the case that the implementation component is somehow able to bypass controls, flout the refusal, and nevertheless execute a prohibited operation (e.g., a payment), even when they are not legally empowered to bring about a particular legal state of affairs with that name. In this case, a prevention by construal mechanism is necessary as the component has no legal ability to bring about payment-in-some-particular-sense, even though they can actually execute something and call it, in some other sense, a payment. The rogue component may succeed in convincing parties outside the normative system that a payment has occurred but will have no such luck in misleading parties that consult the contract database to determine whether a particular (legally recognized) payment occurs. In prevention by *construal*, the contract database ensures that, even though a payment has occurred in some sense, it is not construed as such by a particular provision stored in the database and is therefore of no legal consequence. Forced construals may therefore be used as a preventive mechanism. Here occurrences are regarded as being of a certain type according to a certain clause. It is impossible to coax the occurrence to be of a different type as the database wrapper always appends the clause identifier to the occurrence when it is added to the data store. We saw, in §7.4, how our representation makes use of a function with a restricted domain, buying_function1, to prevent purchases of steel by clerks from being construed as purchases in terms of Clause P.1.

10 Software Implementation

We have developed a software prototype in Java of many of the principles described in this paper. The software, EDEE, currently supports immediate detection (§8.1), bottom-up batched detection (§8.2), and prevention by construal (§9.3). The simple participant-occurrence-role data structure employed by the software enables us to use a broad variety of back-end data stores to uniformly store occurrences, queries, and contractual provisions. EDEE is able to run against Oracle, Postgres, IBM DB/2, Microsoft SQL Server, and Microsoft Access running on SunOS, Red Hat Linux, Windows NT and Windows 2000. The ability to uniformly store queries and contracts in a vendor-independent manner provides greater platform-independence than proprietary stored-procedures and triggers, and potentially allows distribution of contractual provisions across heterogeneous remote sites. With an EDEE wrapper, passive databases—which do not support triggers—can be used for active contract storage and enforcement. Simple active databases can be extended with multi-table triggers that can monitor for complex conditions involving multiple abstract data types, rather than simply detecting updates to a single relation as is conventional [PD94]. Our choice of Java as an implementation approach (over logic programming and theorem proving approaches such as Prolog, PVS, and Isabelle) was driven by a number of factors:

- Java provides platform independence and is standard and pervasively available in industry.
- Query-based reasoning is likely to be more efficient than proof-oriented approaches for large data volumes, as the former takes into account data profiles (predicate selectivity) when executing queries. Optimized query plans can provide substantial performance enhancements on large data sets.
- We saw the need to store rules, and not just facts, in a database, rather than merely leaving rules in unmanageable and non-queryable text files. Further, for conflict detection purposes it is necessary to store, in the database, queryable contents of the rule (queryable occurrence descriptions to analytically check for overlap). Also, for conflict resolution purposes it is necessary to store, in the database, all the attributes of rules (such as author, creation date, unique identifier of provision and its case-based instantiations). Many Prolog implementations do not support rule storage, and we envisaged that storage of and reasoning with rule attributes and contents, and selecting rules to apply in a circumstance, would require as much effort in Prolog as in Java.
- We preferred a set-based semantics (sets of occurrences) rather than a truth-value based approached, and query-based reasoning seemed more suited to union, intersection, count, ordering, and other set-based operations on occurrences.

- Data pre-processors, written in more efficient imperative languages, were in any event necessary to delay inferencing to cope with spikes in update rates (§8.2).

Our system has not yet, however, been tested on actual contracts of any size, and scalability beyond toy examples therefore remains to be demonstrated in an extension of our proof-of-concept implementation. We are satisfied though that the basic mechanisms are in place to store provisions, assess which provisions apply to an occurrence, and analytically detect a range of static and dynamic conflicts through checking for overlap between stored queries. In other work [AB02a,AEB02a], we describe progress towards conflict resolution, though a case-based policy instantiation mechanism which accounts for a norm's life cycle (creation and fulfillment, violation, or voidance) and for defeasible, time-tagged conclusions.

We do not envisage usage of EDEE in an e-market setting, but rather a context where an individual company stores the terms of its contracts in a database and this database is used to guide, and also to conventionally describe, behavior of software and human components within a single organization. Implementation of the software in a distributed setting remains to be addressed; our assumption has been of a single centralized contract database, as we have many issues still to address before moving on to the more complex distributed scenario. Our future work will include improvement of the performance of the coverage determination algorithm. We plan to enhance the interface to the occurrence store by defining high-level business contract definition templates. We also hope to extend our query coverage determination mechanism to be able to support the semantic resolution of hypothetical queries such as 'what obligations is the company subject to if it does not deliver the goods by 26$^{\text{th}}$ July?'.

The following subsections describe in detail how EDEE's coverage-checker operates.

10.1 Storing Operational Data

Let us say SkyHi, a customer of Steelmans, has paid Steelmans £25,000 for a specific shipment. Let `being_supplier1` and `paying1` denote instances (hence the `1` added to create a unique identifier) of occurrences of type *being a supplier* and *paying* respectively. Table 1 shows the occurrence, role, participant schema employed in EDEE to store this operational data.

10.2 Storing Provisions of Contracts and Policies

To store contractual provisions – e.g., "X *prohibits* that [Y be paid]" and "X *promises* that [X pay Y]" – in a relational database we need to handle their embedded propositional content.[13]

[13] See §7 and [AB02c,Kim01].

Table 7. Schemas for storing prohibitions and promises

Occurrence	Role	Participant
prohibiting1	prohibited	Query10
promising1	promised	Query19

(Query10 = occurrences of paying with over £10,000 in role paid ∩ occurrences of paying with a supplier in role payee. See figure 2),
(Query19 = first occurrence of SkyHi paying Steelmans £25,000. See figure 3)

Consider Clause P.3 from the application scenario presented above. We cannot store simply "Steelmans prohibits [paying1]" because paying1 is a concrete instance and might not yet have occurred anyway. We instead store the prohibition as `prohibiting1` in Table 7, and indicate the prohibited occurrences using a pointer to a database view (query) describing the set of prohibited occurrences, which is `query10` in Figure 2. Note that this query would be empty in the case that no prohibited occurrences exist.

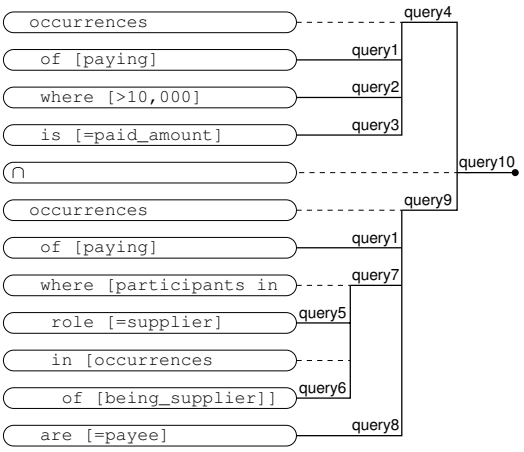
Similarly, the promise in Clause C.1 of our scenario cannot be stored via "SkyHi promises paying1", because we need to store a description of a payment. The promise is thus stored as `promising1` in Table 7 with the promised occurrence represented by the pointer to `query19` in Figure 3. `Query19` asks for the *first* payment since it is exactly one payment that is promised. It may be empty in cases where the promise is broken or voided and no payments are made.

Storing provisions therefore requires the storage of views or queries which describe the promised or prohibited occurrences. Conflicts can be detected from the overlaps between these stored descriptions. The next section describes how the semantics of a query may be stored in a database.

10.3 Storing Queries

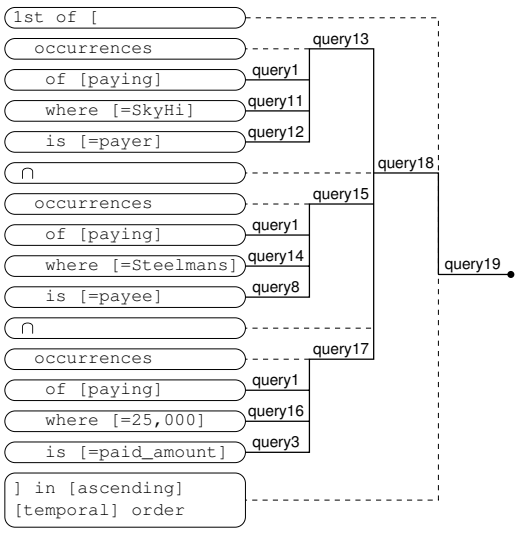
To make queries that return occurrences more concise, we use our own language, EDEEQL [AB02c]. Queries may be stored in occurrence-role-participant tabular form by assigning a query-identifier for each criterion's occurrence entry, and storing its type and value in the role and participant columns respectively. The criterion-value may be constant or a reference to an embedded query. The EDEEQL parser takes the textual form of the query and converts it to its tabular semantic form.

Take for example the query that returns all occurrences where more than £10,000 is paid to a supplier (`query10`, in the `Participant` column, for the row with `prohibiting1`, in Table 7 above). Figure 2 illustrates the parse tree for `query10`, and shows its nested sub-queries (Currency representation is omitted for simplicity). The second query we need to store is "select the first payment of £25,000 by SkyHi to Steelmans" (`Query19` in the `Participant` column, for the row with `promising1`, in Table 7 above). The complete parse



QueryID	Criterion Type	Value
query1	type	paying
query2	greater-than	10000
query3	identified-concept	paid_amount
query4	occurrence	query1
	participant	query2
	role	query3
query5	identified-concept	supplier
query6	type	being_supplier
query7	role	query5
	occurrence	query6
query8	identified-concept	payee
query9	occurrence	query1
	participant	query7
	role	query8
query10	intersectand	query4
	intersectand	query9

Fig. 2. Parse tree and storage schema for query that returns all occurrences where more than £10,000 is paid to a supplier



QueryID	Criterion-type	Value
query11	identified-concept	SkyHi
query12	identified-concept	payer
query13	occurrence	query1
	participant	query11
	role	query12
query14	identified-concept	Steelmans
query15	occurrence	query1
	participant	query14
	role	query8
query16	equal-to	25000
query17	occurrence	query1
	participant	query16
	role	query3
query18	intersectand	query13
	intersectand	query15
	intersectand	query17
query19	set	query18

Fig. 3. Parse tree and storage schema for query that returns the first payment of £25,000 by SkyHi to Steelmans

tree for this query, excluding the repeated query sub-expressions shown earlier, is given in Figure 3. Storing queries explicitly is helpful for finding covering-queries since we can analytically determine which queries, among

a large number of stored queries, cover a certain item or query. We describe the mechanism for finding covering-queries, in the next subsection.

10.4 Finding Overlapping and Inconsistent Provisions

Assume a prohibition, `prohibiting1`, and the associated query describing the prohibited occurrences are stored in an empty database as shown in Table 7 and Figure 2. We then record that Steelmans is a supplier of SkyHi by inserting the occurrence, `being_supplier1`, as described in Table 1 (assume the payment, `paying1`, is not ever inserted). Upon insertion, the coverage-checking algorithm examines each of the unique items `Steelmans`, `supplier`, `SkyHi`, `being_supplier1`, and `supplied` in the set of triples added for this occurrence[14]:

1. **By Rule 4** item `supplier` is covered by the query `[=supplier]`.
2. **By Rule 1** item `being_supplier1` is covered by the query `occurrences of [being_supplier]`.
3. **By Rule 9** the query `occurrences of [being_supplier]` and the query `[=supplier]` dirty the query `[participants in role [=supplier] in occurrences of [being_supplier]]`. Substitution of the input dirt for the dirtied criteria (shown underlined) yields the partial re-evaluation query: `[participants in role [=supplier] in [=being_supplier1]]`. Evaluation of this query yields the output dirt `Steelmans`.
4. **By Rule 9 and step 3** Item (`Steelmans`) dirties query `occurrences of paying where [participants in role [=supplier] in occurrences of [being_supplier]] are [=payee]`. Substitution of the input dirt (shown underlined) for the dirtied criterion yields the partial re-evaluation query: `occurrences of paying where [=Steelmans] are [=payee]`. Evaluation of this partial re-evaluation query yields no output dirt. The coverage-checker thus stops.

We conclude that the new occurrence, `being_supplier1`, is not prohibited, since the only query that covers it is, `occurrences of [being_supplier]`, which is not in the `prohibited` role in any prohibition. We nevertheless record which queries were dirtied by this new data and cache the output dirt, since we can use this dirt in future partial re-evaluations. The dirtied queries and their output dirt is shown in Table 8.

The cache of dirtied queries facilitates creation of more specific partial re-evaluation queries. Even if the actual dirt cache was cleared to conserve resources, we can still rely upon the query optimizer to only re-evaluate the minimal set requiring re-evaluation.

Say SkyHi promises to pay to Steelmans £25,000. Assume this payment has been contemplated, but not effected; no occurrence of `paying` has been added to the data store. As shown in Table 7 and Figure 2, the promise can

[14] Each of the rules mentioned here is defined in detail in the Appendix.

Table 8. Dirtied queries and their output dirt after addition of occurrence of `being_supplier1` to a new data-store

Dirtied Query	Output Dirt
query5	supplier
query6	being_supplier1, ...
query7	Steelmans

(query5 = [=supplier]), (query6 = occurrences of [being_supplier]),
(query7 = [participants in role [=supplier] in occurrences of [being_supplier]])

be represented by embedding the stored query, `query19`, in an occurrence of *promising*. Now, comparing the description of the promised occurrences (`query19`) to other stored queries, proceeding from its most deeply nested sub-expressions upwards:

1. **By Rule 3** `[=25,000]` (`query16`) is covered by `[>10,000]` (`query2`)
2. **By Rule 7** `query4`, which is `occurrences of paying where [>10,000] is [=paid_amount]` covers the query `occurrences of paying where [=25,000] is [=paid_amount]` (`query17`)
3. **By Rule 5** `[=Steelmans]` (`query14`) is covered by any query which covers `Steelmans`. As seen earlier, `[participants in role [=supplier] in occurrences of [being_supplier]]` (`query7`) covers `Steelmans`. This fact is stored in the last row of the "dirtied query and dirt" cache shown in Table 8. Therefore `query7` covers `query14`.
4. **By Rule 7, and step 3** `occurrences of paying where [=Steelmans] is [=payee]` (`query15`) is covered by the query `occurrences of paying where [participants in role [=supplier] in occurrences of [being_supplier]] are [=payee]` (`query9`).
5. **By Rule 6, step 2 and step 4** `Query18` is covered by `Query10`
6. **By Rule 8 and step 5** The `set` criterion (`Query18`) covers `Query19`
7. **By Rule 2, step 5 and step 6** `Query19` is covered by `Query10`.

We have thus shown that what is promised (the description of the promised occurrences = `Query19`) in this context is covered by what is prohibited (the description of the prohibited occurrences = `Query10`). We have thus detected a dynamically appearing *conflict* between a provision embedded in a contract, and an organizational policy. In [Abr02], [AEB02a], and [AB02a], we describe mechanisms for resolving such conflicts. [Abr02] gives an experimental evaluation of the performance of our dynamic conflict detection mechanism. Research into increasing the efficiency of our implementation is ongoing.

11 Related Work

Computational aspects of Kimbrough's approach are dealt with in Daskalopulu and Sergot [DDM01]. Their Prolog implementation models the propositional content inside the disquotation brackets as a prototypical, hypothetical, identified occurrence. These hypothetical occurrences are then matched to actual occurrences to determine if a promise is kept; this is in many respects analogous to determining if an obligation is fulfilled. The developer is required to state separate matching rules for each new object type defined, as well as for occurrences with non-standard thematic roles. The number of matching rules is then linear with the number of object types. In contrast, our query-based approach provides coverage checking (matching of objects to queries which cover them) as a standard system service. We make use of a small number of rules to determine whether an object of any type matches a query, and no additional rules are required as new object types are added. The query-based approach is also well-suited to the specification of free-choice obligations such as the obligation to 'deliver a thick-base pizza or a burrito with extra cheese'. Such an obligation is awkward for an approach based on matching prototypical objects: it is unclear what the prototypical object in this case is, since pizzas and burritos are different object types entirely. A query based approach simply states the query as `first of ((delivering a thick-base pizza) union (delivering a burrito with extra cheese))`. As any appropriate delivery can fill this query, the obligation is straightforward to express.

Grosof, Labrou, and Chan [GLC99] aim to declaratively represent and execute contracts using Situated Courteous Logic Programs. No representation of obligations, permissions, and powers is provided; their contracts are simply labeled if-then rules with procedural attachments and rule override facilities. Theirs is a synchronous approach where procedures are fired during deduction, whereas ours is an asynchronous approach where components look up and fulfil their obligations. Rule labels are manually assigned and non-unique in their approach, requiring the developer to refer to the rule by label and forcing relabeling in cases when new rule conflicts are discovered. Our clause identifiers are system-assigned and unique, and our provisions can be referred to either by label (clause-id) or by a query which specifies the attributes of the provision. Their rules are held in text files; our provisions are managed in a database. Furthermore, as we explicitly express obligations, permissions, and powers we are able to automatically detect conflicts at specification time.

Previous software approaches to ascertaining the status of permissions, prohibitions, and obligations have been based on Petri Nets[15] or finite state machines [Das99,DDM01]. Using regular Petri Nets [Lee88a,Das99] for representing workflows has been criticized [Das99] on two points. Firstly, explicit representation of the rights and duties of the parties is obliterated as obliga-

[15] [BLWW95,Das99,Lee88a]

tory, permissible, and prohibited actions are incorporated into the model implicitly rather than explicitly. Secondly, deriving the Petri Net—establishing the procedure—from the business contract is non-trivial. Tagging Petri Nets places with obligations and violation states that hold [BLWW95] helps address the first issue. Our approach, of employing a database representation of provisions and a continuous query mechanism for monitoring, offers some useful advantages over Petri Nets and state machines. An event and state history is stored. New provisions can be added to the database, without requiring re-derivation of a Petri Net or state machine, which is a difficult manual task. Expressing provisions which entail that multiple obligations are brought about by separate occurrences is easily achieved through a function device; a problem not tackled in Petri Nets and state machines which deal with provisions referring to single obligations. Assessing which obligations persist in a successor state to a state in which multiple obligations applied is easier in a database approach than in a finite state machine or Petri Net approach where multiple graphs would be required and interactions between such graphs are difficult to manage.

A substantial body of literature on the logic of obligations, prohibitions, and permissions (deontic logic) exists; see, for example, Meyer and Wieringa [MW93]. Formal deontic logics have provided valuable insights into the detection of conflicting norms. Our aim has been to enlighten the extension of theory to practice, and, in so doing, contribute to improvement of the theory. In Standard Deontic Logic (SDL) the inter-definition of obligation, prohibition, and permission excludes the possibility of conflicting norms; an assumption we see as too strong in practice, where conflicts are unavoidable. Kimbrough's approach quantifies over obligations, permissions, and prohibitions rather than treating them as operators as in SDL. Individual obligations are therefore identifiable, conflicts are expressible, and each obligation instance can then have a life cycle extending from creation to fulfillment, violation, or voidance. SDL neither captures the provenance of a norm (e.g., its author, specification time, or document position), nor a unique identifier for each norm. We believe this makes conflict resolution for practical applications untenable as insufficient information about provision instances exist to enable choice amongst them.

12 Conclusions

Our contribution has been to describe a software implementation of an updated version of Kimbrough's formal Disquotation Theory. We made a number of revisions to the framework in the process: provisions were relativized to an utterance (clause), rather than to the norm system as a whole; violation states and obligation states were treated as separate entities; permissions were extended to include the sense of vested liberty; and a notion of legal power was added. A novel database representation and software wrapper

which mimics Kimbrough's logic was specified using a practical application scenario: we demonstrated how to implement a computational environment for electronic contract modeling, storage, analysis, and execution. An important innovation was employing an incremental continuous query evaluation mechanism for the monitoring of occurrences against contractual provisions stored in tabular data stores, using pervasive industrial technology (Java and relational databases). We specified a number of contract provision monitoring mechanisms—immediate and delayed detection—as well as some contract provision performance and enforcement mechanisms: intervention, prevention by refusal, and prevention by construal. We have proposed a coverage-determination mechanism for queries within e-service environments. We discussed the data and query storage techniques employed by the EDEE system, and through worked examples, demonstrated how our approach efficiently determined conflicts which appeared dynamically between business contracts and organizational policies. Though much remains to be done before we have an industrial strength application, we believe our alterations to the formal logical framework and our novel software prototype implementation have advanced the state-of-the-art in executable specification for e-commerce applications.

13 Acknowledgements

The authors are grateful to Steve Kimbrough, Brian Shand, and the anonymous referees for the very helpful feedback provided on the draft. This research was supported by grants from Microsoft Research Cambridge, the Cambridge Commonwealth Trust, UK Overseas Research Students Scheme, and University of Cape Town Postgraduate Scholarships Office.

A Coverage Checking Rules

Below are the rules used for determining coverage relationships between queries in our example (the complete list is specified in [Abr02]).

Rule 1 An item is covered by queries with matching `type` criteria.
Rule 2 Transitively, a query is covered by any coverer of its coverers.
Rule 3 A numeric equal-to query Q, is covered by an equal-to, less-than or greater-than query if its `equal-to`, `less-than`, or `greater-than` criterion is, respectively, equal to, greater than, or less than Q's `equal-to` criterion. A numeric less-than query Q, is covered by numeric less-than queries where the `less-than` criterion is greater than the `less-than` criterion of Q. A numeric greater-than query Q, is covered by numeric greater-than queries where the `greater-than` criterion is less than the `greater-than` criterion of Q.
Rule 4 Any participant, occurrence, or role is covered by those concept-identification queries whose `identified-concept` criterion is identical to the participant, occurrence, or role identifier.
Rule 5 A concept-identification query is covered by any query that covers its `identified-concept` criterion.
Rule 6 An intersection query Q, is covered by any given intersection query P, if each of P's `intersectands` covers some non-zero number of Q's `intersectands`.
Rule 7 For two participant queries,[16] P covers Q if P's `role` criterion covers Q's and P's `occurrence` criterion covers Q's. Similarly for occurrence queries[17].
Rule 8 An ordinal (sequence) query is covered by its `set` criterion, e.g., `1st [payments]` is covered by `payments`.
Rule 9 A query, Q, dirties any participant or occurrence query that has Q as its `participant` criterion, `occurrence` criterion, or `role` criterion.

References

[AB00] Alan S. Abrahams and Jean M. Bacon, *Event-centric business rules in e-commerce applications*, Workshop on Best Practices in Business Rule Design and Implementation at the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (Minneapolis, MN), OOPSLA, October 2000.

[AB01a] ———, *Event-centric policy specification for e-commerce applications*, Workshop on Policies for Distributed Systems and Networks (Bristol, UK), January 2001.

[16] EDEEQL syntax:
`participant_query = PARTICIPANTS IN ROLE role_criterion IN occurrence_criterion`

[17] EDEEQL syntax: `occurrence_query = OCCURRENCES OF occurrence_criterion WHERE participant_criterion IS | ARE role_criterion`

[AB01b] ———, *Occurrence-centric policy specification for e-commerce applications*, Workshop on Formal Modelling for Electronic Commerce (Norway, Oslo), June 2001.

[AB01c] ———, *Representing and enforcing e-commerce contracts using occurrences*, Proceedings of the 4th International Conference on Electronic Commerce Research, Edwin L. Cox School of Business, Southern Methodist University, Dallas, Texas, USA, November 2001.

[AB01d] ———, *Representing and enforcing electronic commerce contracts over a wide range of platforms using occurrence stores*, Fourth CaberNet Plenary Workshop (Pisa, Italy), October 2001.

[AB02a] ———, *The life and times of identified, situated, and conflicting norms*, Sixth International Workshop on Deontic Logic in Computer Science (DEON'02), Imperial College, London, UK, May 2002.

[AB02b] ———, *A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts*, Group Decision and Negotiations Journal **11** (2002), no. 6, 1–38.

[Abr02] Alan S. Abrahams, *Developing and executing electronic commerce applications with occurrences*, Ph.D. thesis, University of Cambridge Computer Laboratory, 2002.

[AEB02a] Alan S. Abrahams, David M. Eyers, and Jean M. Bacon, *An asynchronous rule-based approach for business process automation using obligations*, Proceedings of the Third ACM SIGPLAN Workshop on Rule-Based Programming (RULE'02) (Pittsburgh, USA), October 2002.

[AEB02b] ———, *A coverage-determination mechanism for checking business contracts against organizational policies*, Proceedings of the Third VLDB Workshop on Technologies for E-Services (TES'02), 2002.

[AEB02c] ———, *Mechanical consistency analysis for business contracts and policies*, Proceedings of the Fifth International Conference on Electronic Commerce Research (Montreal, Canada), October 2002.

[AGMW97] B. Adelberg, H. Garcia-Molina, and J. Widom, *The STRIP rule system for efficiently maintaining derived data*, Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997, pp. 147–158.

[AK96] R. Ayres and P. J. H. King, *Querying graph databases using a functional language extended with second order facilities*, Advances in Databases, Proceedings of the 14th British National Conference on Databases, (BNCOD 14) (Edinburgh, UK), Springer, July 1996, pp. 189–203.

[AK02] Alan S. Abrahams and Steven O. Kimbrough, *Treating disjunctive obligation and conjunctive action in event semantics with disquotation*, Wharton Business School Working Paper Series (2002).

[All95] James Allen, *Natural language understanding*, second ed., The Benjamin/Cummings Publishing Company, Redwood City, California, 1995.

[And58] A.R. Anderson, *A reduction of deontic logic to alethic modal logic*, Mind **67** (1958), 100–103.

[And62] ———, *Logic, norms, and roles*, Ratio **4** (1962), no. 36, 36–49.

[Aus62] John L. Austin, *How to do things with words*, Oxford at the Clarendon Press, Oxford, England, 1962.

[Ben88] J. Bennett, *Events and their names*, Oxford University Press, 1988.

[BLWW95] Roger W.H. Bons, Ronald M. Lee, Renée W. Wagenaar, and Clive D. Wrigley, *Modelling inter-organizational trade procedures using documentary Petri nets*, Proceedings of the Hawaii International Conference on System Sciences, 1995.

[CCS97] L. Cholvy, F. Cuppens, and C. Saurel, *Towards a logical formalization of responsibility*, Proceedings of the Sixth International Conference on Artificial Intelligence and the Law (Melbourne, Australia), ACM Press, 1997, pp. 233–242.

[D+01] N. Damianou et al., *The Ponder policy specification language*, Proceedings of the International Workshop POLICY 2001 (Berlin, Germany) (M. Sloman, ed.), Springer Verlag, LNCS, 2001.

[Das99] Aspassia Daskalopulu, *Logic-based tools for the analysis and representation of legal contracts*, Ph.D. thesis, Department of Computing, Imperial College, University of London, 1999.

[Dav80] D. Davidson, *Essays on actions and events*, Clarendon Press, 1980.

[Dav96] T. Davis, *Lexical semantics and linking in the hierarchical lexicon*, Ph.D. thesis, Stanford University, 1996.

[DB01] T. Dimitrakos and J. Bicarregui, *Towards a framework for managing trust in e-services*, Proceedings of the Fourth International Conference on Electronic Commerce Research, IFIP INFORMS, 2001, pp. 360–381.

[DDLS01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, *The Ponder policy specification language*, Lecture Notes in Computer Science **1995** (2001), 18–38.

[DDM01] A. Daskalopulu, T. Dimtrakos, and T.S.E. Maibaum, *E-contract fulfillment and agents' attitudes*, Proceedings ERCIM WG E-Commerce Workshop on the Role of Trust in E-Business (Zurich), October 2001.

[GLC99] B.N. Grosof, Y. Labrou, and N.Y. Chan, *A declarative approach to business rules in contracts: Courteous logic programs in XML*, Proceedings 1st ACM Conference on Electronic Commerce (EC-99) (M.P. Wellman, ed.), November 1999.

[HBC97] E.N. Hanson, S. Bodagala, and U. Chadaga, *Optimized trigger condition testing in Ariel using Gator networks*, Tech. Report UF-CIS-TR-97-021, CISE Department, University of Florida. Gainesville, FL (USA), November 1997.

[Hoh78] W.N. Hohfeld, *Fundamental legal conceptions as applied in judicial reasoning*, Greenwood Press Publishers, 1978.

[JM00] D.S. Jurafsky and J.H. Martin, *Speech and language processing*, Prentice Hall, 2000.

[JS96] Andrew J.I. Jones and Marek J. Sergot, *A formal characterisation of institutionalised power*, Journal of the Interest Group in Pure and Applied Logic (IGPL) **4** (1996), no. 3, 427–443, Reprinted in [V+97, pages 349–367].

[Kim98] Steven O. Kimbrough, *On ESΘ theory and the logic of the X12 date/time qualifiers*, Proceedings of the Thirty-First Hawai'i International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1998, pp. 330–339.

[Kim01] ———, *Reasoning about the objects of attitudes and operators: Towards a disquotation theory for representation of propositional content*,

Proceedings of ICAIL '01, International Conference on Artificial Intelligence and Law, 2001.

[Kim02] ———, *A note on the Good Samaritan paradox and the disquotation theory of propositional content*, Proceedings of ΔEON'02, Sixth International Workshop on Deontic Logic in Computer Science (John Horty and Andrew J.I. Jones, eds.), May 2002, pp. 139–148.

[Lee88] Ronald M. Lee, *Bureaucracies as deontic systems*, ACM Transactions on Office Information Systems **6** (1988), no. 2, 87–108, Special Issue on the Language/Action Perspective.

[Lin77] L. Lindahl, *Position and change - a study in law and logic*, Synthese Library, vol. 112, D. Reidel, 1977.

[Mak86] D. Makinson, *On the formal representation of rights relations*, Journal of Philosophical Logic **15** (1986), 403–425.

[MS93] J.D. Moffett and M.S. Sloman, *Policy conflict analysis in distributed system management*, Journal of Organizational Computing (1993).

[MW93] J.-J.Ch. Meyer and R. Wieringa (eds.), *Deontic logic in computer science*, Wiley, Chichester, UK, 1993.

[Par90] Terence Parsons, *Events in the semantics of English: A study in subatomic semantics*, Current Studies in Linguistics, The MIT Press, Cambridge, MA, 1990, ISBN: 0-262-66093-8.

[PD94] N.W. Paton and O. Diaz, *Active database systems*, ACM Computing Surveys **1** (1994), no. 31.

[PS97] H. Prakken and M. Sergot, *Defeasible deontic logic: Essays in nonmonotonic normative reasoning*, Synthese Library, vol. 263, ch. Dyadic Deontic Logic and Contrary-to-Duty Obligations, pp. 223–262, Kluwer Academic Publishers, 1997.

[PV00] F. Pianesi and A. Varzi, *Events and event talk: An introduction*, Speaking of Events (J. Higginbotham, F. Pianesi, and A. Varzi, eds.), Oxford University Press, Oxford UK, 2000, pp. 3–49.

[Sow00] John F. Sowa, *Knowledge representation: Logical, philosophical, and computational foundations*, Brooks/Cole, 2000.

[SV85] John R. Searle and Daniel Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, Cambridge, England, 1985.

[TB99] C.P. Thorpe and J.C.L. Bailey, *Commercial contracts*, Kogan Page Limited, 1999.

[V+97] E. Garzòn Valdés et al. (eds.), *Normative systems in legal and moral theory – festschrift for Carlos E. Alchourrón and Eugenio Bulygin*, Duncker & Humblot, Berlin, Germany, 1997.

Legitimacy Checking in Communicative Workflow Design

Aldo de Moor[1] and Hans Weigand[2]

[1] Tilburg University, The Netherlands, AdeMoor@uvt.nl
[2] Tilburg University, The Netherlands, H.Weigand@uvt.nl

Abstract. Communicative workflow modelling is key to describing, analyzing, and designing business procsses in virtual collaborative networks, such as present in e-commerce. To make workflow models meaningful and acceptable to all partners, their legitimacy must be ensured. To this purpose, the underlying norms must be made explicit and used to check model acceptability. A key class of communicative workflow models are captured by our extended workflow loop. Using this loop as the basic unit of analysis, we introduce the concept of workflow loop norms, grounded in, amongst others, internal control theory. Workflow loop schemas are used to represent workflow situations, allowing for actual or proposed situations to be matched with the norms. Using these constructs, we outline our legitimacy checking process for workflow designs, and illustrate it with a case.

1 Introduction

In today's networked organizations, organizational hierarchies are rapidly becoming less relevant for structuring business processes. Intra-organizational, self-organizing teams, learning organizations, and inter-organizational e-business alliances are emerging in which fixed power and communication structures no longer suffice. To make sense of the increasing organizational complexity and dynamics, and to design more adequate supporting information systems, a workflow view on organizational interactions is often helpful. A good example is the increasing prominence of supply chain modelling [KS03]. Thus, workflow modelling is becoming increasingly important as a structured way of describing, analyzing, and designing the collaborative business process.

Many workflow models exist, ranging from Petri-nets representing logistical or production workflows [Aal97] to approaches that capture more of the organizational semantics of business processes [Sch96]. One class of workflow modeling approaches takes a communications view, grounded in the Language/Action Perspective (LAP), as originally introduced by Winograd and Flores [WF87]. In contrast to data-oriented methods such as those based on state-transition or UML interaction diagrams, LAP modeling is based on the notion of *communicative action,* which implies that workflows are seen as grounded in social relationships and focused on organizational coordination.

For example, a request for a certain good not only aims at the performance of a particular action, but is also an action in itself. A successful request creates a commitment for the party that has promised to deliver a service, for instance, thus changing the social world. LAP workflows are represented as *communication loops* between business *roles*. For example, in the Action Workflow approach [MMWFF93] *customers* and *performers* go through four communication acts: in the *preparation*, a customer asks a performer to do something; at the end of the *negotiation* stage, the performer promises to do this; in the *performance* the performer reports that he has done so; and, finally, with the *acceptance* act the customer reports that she is satisfied. In the DEMO (Dynamic Essential Modeling of Organizations) approach [DV98], an *initiator* and an *executor* subsequently (inter)act in similar *order*, *execution*, and *result* stages.

From a coordination perspective, communication loops are more than sequences of communicative acts. LAP imposes a certain normative structure on communication processes [Wd03]. For instance, ActionWorkflow modeling requires communication loops to be complete, or "closed". This means that all stages of a communication loop must be followed, none can be skipped. Another example of a communication norm implicit in, for instance, DEMO, is that the initiator of a transaction must also be the person who evaluates the success of that transaction. It can be argued, however, that, especially in a complex network organization, such evaluative activities can be delegated to third parties. Also, many workflows spawn other workflows, making the coordination of their interdependencies soon very complex. An additional factor is the strong co-evolution of social and technical requirements in the modern organization [Lym96]. The resulting dynamics in organizational requirements, structures, and behaviour greatly increases the complexity of workflow analysis.

In order to ensure successful organizational performance, the workflow specifications of electronic business networks need to be legitimate, implying that they are both meaningful and acceptable to all partners.[1] Such legitimacy is essential to create a sense of trust and ownership in the network and its supporting IT infrastructure. Central to this analysis are the communicative norms that apply to the workflows of a particular community. To ensure the legitimacy of workflow models, it is important that (1) underlying norms are made explicit and (2) actual or proposed workflow models are checked against these norms. However, the complexity of the communicative norms resulting from their subtle definition, compounded by delegation, recursion, and evolutionary aspects, makes manual analysis very hard to perform. In this chapter, we therefore propose a formal approach to the legitimacy checking of communicative workflow loops, as this helps to deal with representational and reasoning complexities.

[1] [dJ01]

In Section 2, we first define the *extended workflow loop*, our basic unit of analysis. Section 3 introduces our concept of workflow loop norms, and shows how they are firmly grounded in internal control theory, among others. In Section 4, we present *workflow loop schemas* as a concise way of representing workflow situations. Section 5 outlines the method for the legitimacy checking of extended workflow loops. We end the article with discussion and conclusions. In the various sections, we illustrate the ideas with a typical case.

2 The Extended Workflow Loop

In [Wd03] we extensively explained extended workflow loops and their norms. In Sections 2 and 3, we give a brief summary of these ideas.

To define the extended workflow loop, we take the *service relationship* between two actors as its starting point. In most cases, this is a symmetric relationship where a *service* or object of value is exchanged against some financial compensation. This is thus a form of a contractual relationship, whether there is a written contract or not. The service has a *provider* and a *beneficiary*, which usually but not necessarily coincides with the *customer* of the service. Service relationships are typically found at the organizational borders, but they may also be explored within organizations.

To activate a service relationship, we start with describing a *service loop*, that is, the communication around the service. This service loop should not be identified with the service relationship, although it necessarily follows from it. From DEMO, we take the workflow loop *roles* of *initiator* and *executor*, to which we add the *evaluator* role. Instead of using the specific DEMO (order, execution, result) and ActionWorkflow (preparation, negotiation, performance, and acceptance) workflow loop phases, we distinguish two, more neutral pairs of communicative acts, each pair defining a conversation: a *request* is followed by a *commit* (an *actagenic* conversation), and a *report* is followed by an *accept* (a *factagenic* conversation). Furthermore, we distinguish three workflow tasks: *initiation* (I), *execution* (X), and *evaluation* (E). The initiation is the preparation of a request. The execution is the actual performance of the service, around which the communication loop revolves. The evaluation or the assessment is the work that must be done before the service report can be accepted. All four communicative acts and three workflow loop tasks are examples of *workflow acts*.

The service relationship is fundamental, but it can be complemented with *delegation* relationships. In this paper, we will focus on delegation on the side of the provider, but delegation at the customer's side is possible as well. Although any workflow loop act can be delegated to an *agent*, the delegating actor, the *principal*, still keeps a responsibility to the customer, that is, the service relationship itself is not delegated. From internal control theory, we derive the distinction in operational and control tasks. We define the functional role of principal to be responsible for the control tasks (initiation and

evaluation), and the agent for the operational (execution part). The control loop is very similar to the service loop, which makes it possible to view them as two types of communication loops.

Figure 1 presents the extended workflow loop model. Notice that the agent has two executor roles, but there is a slight difference between the X-role of the agent in the service loop and the X-role of the agent in the control loop. From a control perspective, the agent's performance may consist of these executive tasks making up the service, but also of the conversations with the beneficiary (so his overall performance in the service loop is what counts as X in the control loop).

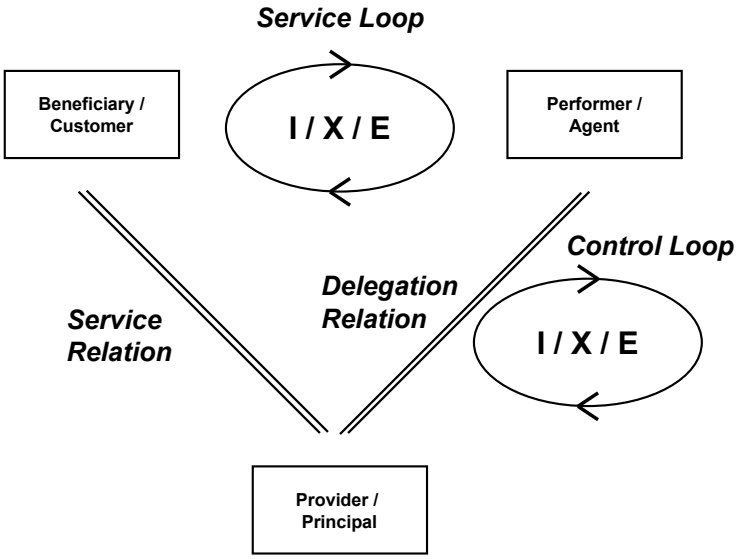


Fig. 1. The extended workflow loop model

3 Extended Workflow Loop Norms

Today's Internet-age information systems are much more communication than computation systems. There are many applications that support complex communication processes, like discussion and group decision making, and many kinds of collaborative work such as group authoring. The semiotics of these systems are often much more complex than in traditional information systems, particularly because the intended semantics and pragmatics are

not under the control of one single organization, and therefore often remain un(der)defined. This entails that often the meaning of information produced and responsibilities for system use and specification are not clear.

3.1 Norms for Organizational Communication

Effective organizational communication presupposes that the communicative partners agree (or can arrive at agreement) on certain norms: not only syntactical norms on the language that is used, but also norms related to the semantics and pragmatics (what are the intended and perceived effects of and on senders and receivers?) of the communication. For instance, a major customer to a company may expect a priority treatment and receive an immediate response to his request for repairs.

Not all organizational communication norms need to be explicit. Many norms, for instance, are implicit or tacit knowledge: everybody is aware of their existence, but they cannot or should not be formalized [Bau99]. For example, in a scientific research group, there is a tacit norm that participants are prepared to defend the claims that they make, whereas a director in a multinational company may expect his staff to follow orders. If such norms are not observed, miscommunications may occur that can have serious effects on the efficacy of the organization.

Having said this, sometimes organizational communication norms do need to be made explicit. They often act as clear decision making rules for staff members. Also, in case of breakdowns [WF87]—for example, by two communicating parties disagreeing on the meaning of a term or responsibility—the rules of action that seemed clear turn out to be really norms that can be violated [OY98].

There are many types of organizational communication norms. One class of norms, for example, concerns social norms defining the effectiveness and efficiency of internal organizational communication aimed at motivating or informing employees [dv96]. Of course, these norms may ultimately influence what are proper workflow loop norms. We, however, do not focus on how workflow norms came to be, but on the norms that (currently) generally apply. Finding such relatively generic principles is important, as these criteria can be built into system designs. By making information systems more legitimate in this way, they can significantly improve organizational communication.

3.2 Workflow Loop Norms Implicit in LAP

One important source of stable workflow loop norms is internal control theory. This theory provides normative guidance in complex organizational structures, when there are delegated task structures which allow agents to establish commitments on behalf of the organization. Delegating an activity does not mean that the responsibility for this activity is delegated as well. Instead,

it introduces a control task for the principal that delegated the task to the agent. This involves communication: since in most cases, the principal responsible for that control task cannot personally observe the performance of operating tasks, he must rely on documentary evidence (evidence function). At the same time, to protect himself, the executing party (the agent) must be able to prove the completion of an activity (preventative function).

An extensive literature related to internal control theory exists, often drawing from accountancy research. For example, Chen [Che92] lists a set of principles such as "An operational task and its corresponding control task should be segregated into two different organizational positions and into two different agents". Bons [Bon97] notes that control tasks can be divided into two categories: control tasks that make direct statements about the operational tasks (such as witness reports), and control tasks that evaluate the resulting document and draw conclusions based on them. Many principles can be found in the literature, but more interesting at the moment is to see how they can actually be used to construct workflow loop norms.

In [Wd03] we proposed an approach to analyze workflows using communicative norms based on such internal control theory norms. We identified the following list as a first approximation of the basic implicit norms underlying LAP:

1. For any activity, a distinction must be made between the *operational task* and the *control* task. These two tasks are executed by different roles and different subjects.
2. If an operational task exists, there should be a corresponding *initiating* control task and the operational task should follow the initiating task.
3. If an operational task exists, its corresponding *evaluative* control task should exist as well and should always follow the operational task.
4. The initiating task should contain a request for action from a role (initiator) independent of the role performing the task
5. The role issuing the initiating task (initiator) should be the same as the role responsible for the (evaluative) control task.
6. The initiating task should be closed with a commitment (promise) from the role performing the operational task.
7. The evaluative control task should be furnished by supporting documents. The supporting documents should originate from the role performing the operational task.
8. The evaluative control task should be closed with a performative statement from the role performing the evaluative control task.
9. The performative statement of the evaluative control task should be received by the role that performed the operational task.

Based on our own extended workflow loop model and principles from internal control theory, we then generalized and refined those implicit LAP norms. We formalized the basic concepts from our extended workflow loops

as well as the norms, resulting in a formal ontology and a set of extended workflow loop norms expressed in terms of this ontology. In the next section, we present an adapted version of this ontology.

3.3 The Extended Workflow Loop Ontology

A normative analysis quickly becomes complex. To deal with this complexity, formal representation and reasoning can be of great help. Having introduced the concepts underlying the extended workflow loop, we are now ready to start our formalization. The formalization consists of two main parts: (1) a *formal ontology* to precisely define the meaning of the extended workflow loop concepts, and (2) a set of *extended workflow loop norms* based on these definitions. We present both the ontology and formal representation in the remainder of this section.

First, we present our extended workflow loop ontology, adapted from [Wd03].

A **service** is a tuple <Service Type, Service Executor, Object, Beneficiary>, where *Service Type* is some predicate designating a service type. The *Object* is the object of value. The object can be immaterial. *Service Executor* and *Beneficiary* are actor roles with respect to the service. The Service Executor role should not be confused with the conversational role of Executor.

A **service relation** is a tuple <Service, Customer, Provider>, where *Customer* and *Provider* are actor roles, and *Service* is some service as defined above.

The **service loop** for a certain service is a tuple <Service, Exec, Init, Eval, Acta, Facta>, where *Exec*, *Init* and *Eval* are tasks and *Acta* and *Facta* are conversations consisting of pairs of communicative acts.

A **delegation relation** is a tuple <Workflow Loop Act, Principal, Agent>, where Workflow Loop Act stands for one or more of the tasks or communicative acts that make up a workflow loop. *Principal* and *Agent* are Actors. For the time being, we omit what exactly has been delegated (that is, task roles and/or conversations). The transitive closure of the delegation relationship is called the *delegation line*.

A **control loop** for a certain delegation relation is a tuple
<Delegation, Exec, Init, Eval, Acta, Facta>,
where Init, Exec, and Eval are acts, and Acta and Facta are conversations. The Exec task of the control loop of a delegation is by definition done by the Agent of the delegation, whereas the Init and Eval acts belong to the Principal, unless he has delegated these acts as well.

3.4 Formal Extended Workflow Loop Norms

The purpose of the ontology is to define precisely the extended workflow loop norms. In [Wd03] we presented a rather comprehensive set of extended workflow loop norms. These formal norms follow from the implicit set listed

in Section 3.2. Here, we take only one of these norms to illustrate the main contribution of this paper: developing an operational method for legitimacy checking of workflow schemas. Other norms can be addressed in similar ways.

XWL Norm 1:

∀s:service_loop(s.init=s.service.customer ∧ s.eval=s.service.customer)

This norm requires that the customer of a service (as defined in the service relation) is both initiator and evaluator – which reflects a strong commitment to customer orientation that may be violated for other reasons.

4 Workflow Loop Schemas

To apply workflow loop norms, we need to distinguish between actual and deontic states. Norms indicate deontic ("soll") states: how the world should (not) be. However, equally important are the current or proposed states on which the norms are to have their regulatory deontic effect. We call these actual states the *workflow situation*. Each workflow situation comprises one or more extended workflow loops. Each extended workflow loop is represented by two workflow loop schemas: a service loop schema and a control loop schema.

4.1 The Schema Structure

In order to model current or proposed *workflow loop situations*, we introduce the concept of *workflow schema*. Schemas can be used to organize knowledge that represents complex situations or objects in a domain.[2] Here, we use a schema to decompose a workflow loop into its constituent acts. To these acts, the various roles and subjects playing these roles are mapped.

Table 1 shows the basic structure of a workflow loop schema. The first row represents the *workflow loop acts*. There are seven possible acts: the init-task (to prepare the request), the request, the commit-act, the execution of the commit, the reporting stage, the evaluation of the results, and the acceptance of the results. As these acts together make up the full workflow loop, they are always completely represented in the schema.

The second row shows the actor roles that carry out the workflow loop acts. These roles are often defined in the specific domain in which the workflows are carried out. The labels are thus domain-dependent.

The third row describes the conversational role (initiator, executor, or evaluator) that performs the workflow loop act. In general, there is a strong

[2] [LS89]

correlation between the workflow loop acts and the conversational roles that perform them. For instance, in DEMO the role performing the evaluative task is always the initiator. However, as our analysis of workflow loop norms has shown, such constraints often need to be relaxed in the complexity of real work situations. In many networked organizations, for example, it is not the initiator, but a separate evaluator role that performs the evaluative task, for reasons of efficiency or to implement checks and balances.

The fourth row captures the complexity introduced in the extended workflow loop: a role can act as a customer, beneficiary, provider, principal, or agent. As many mappings to the other roles are possible, a separate row is justified. For example, in case of delegation, the customer role requesting a service is not necessarily the beneficiary.

The fifth row describes the subjects performing the workflow loop acts. In DEMO, for example, no specific constraints are imposed on which subjects perform the acts. From an internal control perspective, however, many constraints (i.e., prohibitive norms) are often demanded, such as that the evaluation of performance cannot be done by the same subject who has executed the work.

4.2 Modelling Workflow Loop Situations

In Table 1, we use the workflow loop schema to model a simple workflow loop situation: a baker promises his customer to bake a bread, upon request, and then bakes and delivers the bread himself. In this case the customer explicitly asked the baker to bake the bread. No preparation of the request was needed, so the init-task remains empty. This situation has been modelled using standard LAP semantics, with no explicit evaluator role distinguished. As there is no delegation, only the provider and customer[3] roles need to be defined. It is clear that in this situation there are only two subjects performing the subsequent workflow loop acts. For clarity, these subjects have labels here similar to the domain roles that they play. However, normally there is not necessarily a one-to-one mapping between subjects and domain roles, as one subject can play more than one such role, for example.

The simple workflow loop can be modelled using a single workflow loop schema as follows:
Note that there are two customer-roles here: a domain role (somebody buying a bread) and a workflow loop role (the LAP role).

To illustrate how workflow loop schemas can also be used to model the more complex workflow situations typical of extended workflow loops, we

[3] Alternatively, or additionally, the *beneficiary* role could have been modelled. However, current LAP semantics is unclear about the precise differences. We therefore limit ourselves to the customer role here. Note that structuring semantics in these workflow loop schemas can be of considerable help in identifying such semantic unclarities or gaps.

Table 1. Using a workflow loop schema to represent a simple workflow situation

Workflow loop: Simple Pizza Delivery							
WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	-	Cust.	Baker	Baker	Baker	Cust.	Cust.
Conv. Role	-	I	X	X	X	I	I
WL Role	-	Cust.	Prov.	Prov.	Prov.	Cust.	Cust.
Subject	-	#cust	#bak	#bak	#bak	#cust	#cust

now represent the scenario about the extended pizza delivery case described in [Wd03].

> *The Complex Pizza Delivery Case*
>
> In the case of a pizza baker who originally bakes and delivers his pizzas himself, the communication between him and a customer can be easily modelled using a standard workflow loop (within a contract relation, but we will focus here on the baker as performer). Now the baker hires a boy to deliver the pizza to the house of the hungry client for him. Then there exists an agency relation between the baker and the boy: the baker plays the manager/principal role, the boy the employee/agent role. The workflow loop now seems distorted, since the new pizza delivery workflow loop performer is no longer one subject. Say the hungry client calls the baker on the phone. In an actagenic conversation, part of the workflow loop, the baker agrees to bake and deliver a pizza. After calling the boy, the baker orders the boy to bring the pizza to the client. The boy takes the pizza, drives to the house, rings and starts a factagenic conversation in which the hungry client accepts the pizza, perhaps after having signed a note. The boy returns to the baker and reports the succesful delivery, possibly with handing over the note as evidence.

Note that there are now a control loop and a service loop, together forming the extended workflow loop. We therefore require two—related—workflow loop schemas to represent the complex workflow loop norms (including delegation) implicit in this case. See Table 2.

These tables should be mostly self-explanatory after the previous introduction. Note that there is now an Init-task (namely, the baking of the pizza) which has to be done before the baker can request the boy to deliver the pizza. Also, the workflow loop roles in the service loop schema are distributed among customer, provider, and agent. In the example, we abstain from the difficulties added by more than one subject playing a particular role of some act. Such set-theoretical issues need to be addressed in future work though.

Table 2. Using workflow loop schemas to represent a complex workflow loop situation

Service Loop Schema: Complex Pizza Delivery							
WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	-	Cust.	Baker	Boy	Boy	Cust.	Cust.
Conv. Role	-	I	X	X	X	E	E
WL Role	-	Cust.	Prov.	Agent	Agent	Cust.	Cust.
Subject	-	#cust	#bak	#boy	#boy	#cust	#cust

Control Loop Schema: Complex Pizza Delivery							
WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	Baker	Baker	Boy	Boy	Boy	Baker	Baker
Conv. Role	I	I	X	X	X	E	E
WL Role	Princ.	Princ.	Agent	Agnt	Agnt	Princ.	Princ.
Subject	#bak	#bak	#boy	#boy	#boy	#bak	#cust

5 A Method for Legitimacy Checking

The basic idea underlying the method is that the extended workflow loop schemas, as well as the norms, can be defined as semantic networks in the form of conceptual graphs. The workflow loop norm graphs put selectional constraints on the workflow schema graphs, in other words, on the actual or proposed workflow loop situation. These norms define what schema elements are required or forbidden. Legitimacy is checked by projecting the workflow norm patterns on the workflow schema definitions. If required patterns have are matched (i.e. have projections), and forbidden patterns have no matches (i.e. have no projections), no norm violations occur. The present set of workflow schema definitions, and thus of the workflow situation, is thus legitimate. Next, we first formalize the workflow schemas using conceptual graphs. We then present our method for legitimacy checking.

5.1 Conceptual Graph Theory

In Section 3 we formalized the norms by defining an ontology and norms using first order logic. We now formalize the concept of schema, in order to provide precise semantics and be able to reason about their properties. To this purpose, we use conceptual graph theory. Two important advantages of conceptual graphs are that they allow for the efficient construction of generalization hierarchies of graphs, and that they can represent contextualized or nested definitions. These properties are needed to efficiently check normative knowledge definitions. We will give a brief introduction of conceptual graph theory, as it is relatively unknown. The theory is explained in much more detail in [Sow84].

Conceptual graphs are constructed out of concepts and relations.

Concepts. A *concept* has two fields: a type and a referent field. It has the following format:

[*Type*: *Ref*], an example being [Customer: #John]

The type field contains a type label that is part of a type hierarchy. The referent field designates a particular entity with the mentioned type. This field is optional: if not specified, the concept is considered to be generic, and is by default existentially quantified.

A referent can be an individual marker or a generic marker. An individual marker can be a number sign, followed by some constant, e.g., #John. A generic marker, denoted by *, indicates a generic concept. It may be followed by a variable identifier, e.g., *x1. This allows one to refer to a specific, but as of yet unidentified entity. These named generic markers are useful for cross-referencing concepts in graphs. A co-referent concept is indicated with a question mark, e.g., ?x1.

Conceptual Relations. A conceptual relation links two or more concepts. Each conceptual relation has a relation type, surrounded by parentheses. It also has one or more arcs, represented by arrows, each of which must be linked to some concept. A dyadic relation has the following representation:

[$Type_1$: Ref_1] -> (*R_Type*) -> [$Type_2$: Ref_2]

Generally, the relations can be read, in the direction of the arrows, as 'the *source concept* has a *relation* to the *destination concept*'. An example of a conceptual relation could be:

```
[XWL: #Pizza_Delivery] ->
(Part) -> [Service_Loop]
```

which states that the extended workflow loop for pizza delivery has some (yet unspecified) service loop.

Conceptual Graphs. A conceptual graph is a combination of concept nodes and conceptual relation nodes. It can also consist of a single concept node. To represent such a graph, one of its concepts is chosen as its head. If more than one relation is linked to a concept, the dash symbol '-' can be used to separate the common concept from the rest of these relations. A conceptual graph is ended by a period. Many examples of these graphs are given in the remainder of this section.

5.2 Outline of the Method

The method for legitimacy checking of workflow loop norms is an adaption of the method used in [dv01] to match required and enabled web service functionality.

The method consists of the following steps:

1. Define an extended workflow loop ontology
2. Represent the workflow loop norms in patterns
3. Represent the workflow situation in workflow loop schemas
4. Calculate the match between situations and norm patterns
5. Interpret the matching results

Define the Extended Workflow Loop Ontology. First, we define the extended workflow loop ontology as a type hierarchy, with accompanying type definitions. All entities followed by a '>' sign are supertypes of the indented entities that follow. The hierarchy and type definitions follow from the previous discussion.

The Extended Workflow Loop Type Hierarchy.

```
Entity >
     WL_Act >
          Task >
               Init
               Exec
               Eval
          Comm_Act >
               Request
               Commit
               Report
               Accept
     Actor >
          Serv_Actor >
               Beneficiary
               Serv_Executor
          Dom_Role
          Conv_Role >
               Inititiator
               Executor
               Evaluator
          WL_Role >
               Customer
               Provider
               Principal
```

```
          Agent
Conversation >
     Acta
     Facta
Comm_Loop >
     Service_Loop
     Control_Loop
Deont_Eff >
     Perm
     Req
     Forb
Relation >
     Del_Rel
     Serv_Rel
Schema >
     Comm_Loop_Schema >
          SL_Schema
          CL_Schema
Object
Service
Speech_Act
Subject
XWL
```

There is also a short relation type hierarchy, defining such roles as `(Agnt)`, `(Part)`, and (`Conv_Role`), but this will be omitted here, as it is a flat tree, with no relation subtypes.

The semantics of the concept types are given by the following type definitions:

- Workflow loops revolve around the performance of services. A service is of a particular type, has some object, and is done by an executor for a beneficiary. As in conceptual graphs, the type is already mentioned in the concept, no separate Service Type concept is needed.

```
[Service: *x] -> (Def)-> [Entity: ?x] -
     (Obj) -> [Object]
     (Agnt) -> [Serv_Executor]
     (Ptnt) -> [Beneficiary]
```

Notice that such a (meta) type definition indicates the ontological properties that a concept *must* have. This definition may be contracted, by replacing the genus (entity) by the defined type (service) and dropping the (Def) relation.

- An extended workflow loop consists of both a service and a delegation relation as well as a service loop and a control loop.

```
[XWL: *x] -> (Def)-> [Entity: ?x] -
     (Part) ->  [Serv_Rel]
     (Part) ->  [Del_Rel]
     (Part) -> [Service_Loop]
     (Part) -> [Control_Loop]
```

- Each extended workflow loop contains two relations: a service relation and a delegation relation.

```
[Serv_Rel: *x] -> (Def) -> [Relation: ?x] -
       (Obj) -> [Service]
       (Agnt) -> [Customer]
       (Agnt) -> [Provider]

[Del_Rel: *x] -> (Def) -> [Relation: ?x] -
       (Obj) -> [WL_Act]
       (Agnt) -> [Principal]
       (Agnt) -> [Agent]
```

- Both a service loop and a control loop are communication loops. Each of these loops has a communication loop schema. This consists of an Init-task, a Request-act, ... The Init-task is done by a domain role, a conversation role, a workflow loop role, and subject.

```
[Comm_Loop_Schema: *x] ->
  (Def) -> [Schema: ?x] -> (Part) -
       [Init] -
            (Dom_Role) -> [Dom_Role]
            (Conv_Role) -> [Conv_Role]
            (WL_Role) -> [WL_Role]
       [Request] -
            ...
       [Commit] -
            ...
       [Exec]
            ...
       [Report]
            ...
       [Eval]
            ...
       [Accept]
           ...]
```

- Each communication loop contains two conversations, an actagenic and a factagenic conversation. Although for completeness we define its semantics, we do not further address how conversations and larger communicational structures can be used here.

```
[Acta: *x] -> (Def) ->
     [Conversation: ?x] -> (Part) -
       [Request]
       [Commit]

[Facta: *x] -> (Def) ->
     [Conversation: ?x] -> (Part) -
       [Report]
       [Accept]
```

Represent the Workflow Loop Norms in Patterns. To represent the norms we define two kinds of patterns: *required patterns* and *forbidden patterns*. These patterns are projected on the workflow schemas, to check whether they conform to the workflow loop norms. Thus, these patterns act as meta-norms, norms about norms, in this case stating whether patterns must or may not be there, respectively [LSDN01].

Each workflow loop norm can be translated into one or more required and forbidden patterns. Here we show this for the first extended workflow loop norm (XWL Norm #1). This norm said that *the customer of a service must be both the initiator and the evaluator of the accompanying service loop.* Ontological concepts that are relevant here are: the service relation, service loop, customer, initiator, and evaluator. This norm is translated into the following patterns:

Required patterns. For the customer #cust of the service (as defined by the service loop), there is only one required pattern #rp1.

```
[SL_Schema] -> (Part) -
     [WL_Act] -
          (WL_Role) -> [Customer: #cust]
     [WL_Act] -
          (Conv_Role) -> [Initiator: #cust]
     [WL_Act] -
          (Conv_Role) -> [Evaluator: #cust]
```

For all service loop schemas, the customer, initiator, and evaluator roles must be played by the same subject #cust.

Forbidden patterns. As according to XWL Norm #1 the customer must be both *the* initiator and *the* evaluator, there may not be schemas in which the either one of these roles is played by a subject other than #cust. Thus, the following two patterns #fp1 and #fp2 are forbidden:

```
[SL_Schema] -> (Part) -
   [WL_Act] -
      (Agnt) -> [Initiator: *y1] where
```

```
        *y1 ∉ #cust

[SL_Schema] -> (Part) -
   [WL_Act] -
      (Agnt) -> [Evaluator: *y1] where
         *y1 ∉ #cust
```

Represent the Workflow Situation in Workflow Loop Schemas. Key to the definition of the workflow situation are the workflow schemas. Auxiliary definitions, such as a list of the services and service relations also need to be defined, but will be omitted here.

There are two workflow loop schemas, each of which is a specialization of the workflow loop type definition given in the previous section. Note that in this case we have not represented domain roles, as they are not relevant in this case. In fact, *the* baker, *the* boy and *the* customer are regarded as subjects only. Instead of using the abstract subject notations #s1, #s2, and #s3 we use the more comprehensible subject identifiers #customer, #baker, and #boy in the graphs. However, in other especially organizational, cases, the domain role is indeed important. Many formal norms, for example, are domain-dependent: *a* manager may, *an* employee must, etc. In future work, we will investigate the role of this additional complexity in our schemas.

The service loop schema of the case #sl1 is:

```
[SL_Schema: #Complex_Pizza_Del] -> (Part) -
     [Request] -
          (Conv_Role) -> [Initiator: #cust]
          (WL_Role) -> [Customer: #cust]
     [Commit] -
          (Conv_Role) -> [Executor: #bak]
          (WL_Role) -> [Provider: #bak]
     [Exec] -
          (Conv_Role) -> [Executor: #boy]
          (WL_Role) -> [Agent: #boy]
     [Report] -
          (Conv_Role) -> [Executor: #boy]
          (WL_Role) -> [Agent: #boy]
     [Eval] -
          (Conv_Role) -> [Evaluator: #cust]
          (WL_Role) -> [Customer: #cust]
     [Accept] -
          (Conv_Role) -> [Evaluator: #cust]
          (WL_Role) -> [Customer: #cust]
```

The control loop schema of the case #cl1 is:

```
[CL_Schema: #Complex_Pizza_Del] -> (Part) -
     [Init] -
          (Conv_Role) -> [Initiator: #bak]
          (WL_Role) -> [Principal: #bak]
     [Request] -
          (Conv_Role) -> [Initiator: #bak]
          (WL_Role) -> [Principal: #bak]
     [Commit] -
          (Conv_Role) -> [Executor: #boy]
          (WL_Role) -> [Agent: #boy]
     [Exec] -
          (Conv_Role) -> [Executor: #boy]
          (WL_Role) -> [Agent: #boy]
     [Report] -
          (Conv_Role) -> [Executor: #boy]
          (WL_Role) -> [Agent: #boy]
     [Eval] -
          (Conv_Role) -> [Evaluator: #bak]
          (WL_Role) -> [Principal: #bak]
     [Accept] -
          (Conv_Role) -> [Evaluator: #cust]
          (WL_Role) -> [Principal: #cust]
```

Note that—contrary to the service loop—the Init-task is now performed by the baker: baking the bread is a necessary preparatory act for the baker to be able to request the boy to deliver it. The Eval-task of the baker could, for instance, consist of regularly checking with the customer if the deliveries are in time, either face-to-face in the shop or by phone.

Calculate the Match between Workflow Loop Schemas and Norm Patterns. S is the set of all workflow loop schemas. RP is the set of all required patterns, FP is the set of all forbidden patterns.

S = { #sl1, #cl1 }, RP = { #rp1 }, FP = { #fp1, #fp2 }.

– Project all required patterns on all workflow loop schemas s ∈ S:

RM = the set of schemas matching (i.e. being a specialization of) *all* required patterns.

Here RM = { SL_Schema: #Complex_Pizza_Del }

– Project all forbidden patterns on all workflow loop schemas s ∈ S:

FM = the set of schemas matching *any* of these patterns. Here FM = ∅.

Interpret the Matching Results. If ∀s ∈ S: s ∈ RM and s ∉ FM, then the workflow situation is legitimate, otherwise it is illegitimate. In that case, the conflicting pattern(s) must be dealt with by redefining one or more of the workflow loop definitions. The proposed definitions must be checked in turn

by running the calculation again. How exactly this interpretation process is to occur, depends on (1) the type of workflow loop definitions causing the violation, (2) the type of norm being violated, and (3) the meta-norms governing what should be done in case of violation. No uniform interpretation approach can thus be given. In future research, we intend to develop interpretation classifications to structure such norm conflict resolution processes.

6 Conclusions

In contemporary information society, the quality of the communication in and between organizations is becoming a critical success factor. To design and maintain effective communication systems, we need more than communication modeling. We should also be able to check the quality of the communication models. Thus, there must be norms that define legitimate, i.e. meaningful and acceptable organizational communication.

In modern rational organizations and networks, not only the communication structures themselves must be legitimate, but also the process in which these norms are generated. This means that communication norms may have to be made explicit, and become the subject of a rational discussion.

These two considerations provided the motivation for this chapter on the role of legitimacy checking in communicative workflow loop design. First, we have shown an analysis of communication norms based on the Extended Workflow Loop model. Using this model to define a workflow loop ontology and accompanying norms, we have described a practical method of legitimacy checking. The method uses the notion of workflow loop schemas in which various elements of the communication workflow loops are integrated. It is shown how such a schema can be represented using Conceptual Graph Theory. This makes it possible to delegate the norm checking to reasoning tools. Note that the norm checking process itself is not yet necessarily legitimate; for this it should be embedded in a social process in which the communication structures and norms can be discussed and challenged by relevant stakeholders.

One important application of the legitimacy checking method is communication diagnosis [vWd02], which works in a bottom-up fashion. The goal of diagnosis is to model the current situation and to analyze actual or potential flaws by linking them to communicative norm violations. The diagnosis should result in recommendations for improvement. In the case of workflow redesign, the reengineering process description should indicate how the new situation can be reached from the current situation by redefining workflow structures that violate the communication norms. This reengineering process itself must also be legitimate.

The current trend in information system development is a move away from detailed and formal methodologies [AF03]. Formalization is not a goal in itself; what is needed is rationalization. A more contingent approach is therefore needed. What is most problematic in contemporary, elaborate methodologies,

is, in our view, the lack of attention to systematic involvement of users—as stakeholders—in collaborative system (re)design. Admittedly, user involvement and stakeholder analysis have been gaining prominence for a long time. Some approaches use brainstorming sessions, for instance. However, this is still far off from encouraging rational discussion. A rational discussion also allows participants to challenge the norms that underly the design choices. A legitimacy checker as discussed in this paper can be instrumental in such a process and—if embedded in a carefully designed social interaction process—can be an example of a useful application of formal methods.

References

[Aal97] W.M.P. van der Aalst, *Three good reasons for using a Petri-net-based workflow management systems*, Information and Process Integration in Enterprises: Rethinking Documents (Toshiro Wakayama, Srikanth Kannapan, Chan Meng Khoong, Shamkant Navathe, and JoAnne Yates, eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1 December 1997, pp. 179–201.

[AF03] David E. Avison and Guy Fitzgerald, *Where now for development methodologies?*, Communications of the ACM **46** (2003), no. 1, 78–82.

[Bau99] P. Baumard, *Tacit knowledge in organizations*, Sage, London, UK, 1999.

[Bon97] Roger W.H. Bons, *Designing trustworthy trade procedures for open electronic commerce*, Ph.D. thesis, EURIDIS at Erasmus University, Rotterdam, The Netherlands, September 1997.

[Che92] K. T. Chen, *Schematic evaluation of internal accounting control systems*, Ph.D. thesis, University of Texas at Austin, Austin, Texas, 1992.

[dJ01] Aldo de Moor and Manfred A. Jeusfeld, *Making workflow change acceptable*, Requirements Engineering **6** (2001), no. 4, 75–96.

[dv96] B. de Wit and M. van Delden, *Performance analysis of internal communication: A research approach for assessing the quality of policy and motivating communication*, The Quality of Communication in Organisations in Theory and Practice (J. A. de Ridder and K. Seisveld, eds.), Cramwinckel, Amsterdam, The Netherlands, 1996, Translated from Dutch.

[DV98] J. Dietz and V. Van Reijswoud, *DEMO modelling handbook*, Technical report, 1998, Volume 2.

[dv01] Aldo de Moor and W. J. van den Heuvel, *Making virtual communities work: Matching their functionalities*, Proceedings of the 9th International Conference on Conceptual Structures (Palo Alto, CA), Stanford University, July 30–August 3, 2001.

[KS03] D.C.L. Kuo and M. Smits, *Performance of integrated supply chains: An international case study in high tech manufacturing*, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36) (Ralph H. Sprague, Jr., ed.), IEEE Computer Society Press, Los Alamitos, CA, January 2003, (CD-ROM).

[LS89] G. Luger and W. Stubblefield, *Artificial intelligence and the design of expert systems*, Benjamin-Cummings, Redwood City, CA, 1989.

[LSDN01] K. Liu, L. Sun, A. Dix, and M. Narasipuram, *Norm-based agency for designing collaborative information systems*, Information Systems Journal **11** (2001), no. 3, 229–247.

[Lym96] P. Lyman, *How is the medium the message? notes on the design of networked communication*, Computer Networking and Scholarly Communication in the Twenty-First Century University (T. Stephen, ed.), State University of New York Press, New York, NY, 1996, pp. 39–52.

[MMWFF93] Raul Medina-Mora, Terry Winograd, Rofrigo Flores, and Fernando Flores, *The action workflow approach to workflow management technology*, The Information Society **9** (1993), no. 4, 391–404.

[OY98] Wanda Orlikowski and JoAnne Yates, *Genre systems: Structuring interaction through communicative norms*, Working paper CCS WP #205, Sloan WP #4030, Massachusetts Institute of Technology, Sloan School of Management, Cambridge, MA, July 1998, http://ccs.mit.edu/papers/CCSWP205/index.html, accessed 22 August 2004.

[Sch96] T. Schäl, *Workflow management systems for process organizations*, Springer Verlag, June 1996.

[Sow84] John F. Sowa, *Conceptual structures: Information processing in mind and machine*, Addison-Wesley, Reading, MA, 1984.

[vWd02] F. van der Poll, H. Weigand, and A. de Moor, *Communication diagnosis of a financial service process*, Proceedings of the Seventh International Workshop on the Language-Action Perspective on Communication Modelling (Delft, The Netherlands), LAP-2002, June 12–13, 2002, pp. 118–134.

[Wd03] Hans Weigand and Aldo de Moor, *Workflow analysis with communication norms*, Data & Knowledge Engineering **47** (2003), 349–369.

[WF87] Terry Winograd and Fernando Flores, *Understanding computers and cognition: A new foundation for design*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part I: Syntax & Formal Semantics of CANDID

Ronald M. Lee

Florida International University (FIU), Miami, FL, USA, r.lee@fiu.edu

Abstract. The formal language CANDID is presented as a knowledge representation formalism for artificially intelligent decision support systems. The language is specifically oriented to representation of concepts in finance, commerce and administration. Later parts of the paper demonstrate the application of CANDID to explication of corporate entities and contractual objects, as well as to various concepts in elementary finance.

1 Introduction

This part presents the syntax and formal semantics of the language we have called CANDID, originally described in Lee [Lee80].

In the discussion that follows, the reader is presumed to be familiar with the first order predicate calculus (FOPC), which we takes as our starting point. For background, we suggest the text by Kalish, Mantague and Mar [KMM80]. The extension to this that comprise CANDID are drawn chiefly from Montague's "intensional logic" [Mon02,Dow78], and Von Wright's "deontic logic",[1] with minor influence from the temporal logic of Rescher and Urquhart [RU71]. The presentation given here is a model theoretical one. Background on model theory is given in Dowty [Dow78] Kalish et al. [KMM80] mentioned above. Deeper coverage is provided in van Fraasen [vF71] and Chang and Keisler [CK73].

The CANDID language as described here loosely follows the develop of Montague's Intensional Logic, as presented in Dowty [Dow78], augmented with the operators of Von Wright's Deontic Logic. The principle differences up to the language IL (Intensional Logic) are as follows:

- addition of operators and the definite reference operator ι.
- omission of the tense, P and F (past and future)
- addition of the sets C (character strings) and N (numbers) in the model.

[1] [VW65,VW67,VW68]

- recognition of time (designated as the set T rather than J) with the object language; addition of the operator R for temporal realization (adapted from a similar notation by Rescher and Urquhart.[2]

The language IL is then extended to include the connectives and operators of Von Wright's deontic logic with the following modifications:

- addition of an agent place in the I connective
- re-interpretation of contingent permission and obligation
- addition of operators for contractual obligation and permission, and the connective OE ("or else").

1.1 General Notational Conventions

Throughout this paper we will describe a series of formal languages of increasing complexity. The formal language itself will be called the *object language*, whereas its description is done via a *meta-language*.

Object Language – Constants In the object language, constant names will be strings of upper or lower case Roman letters or digits or dashes, beginning with a capital letter. These will denote individuals in the domain. Later, the object language is extended to include symbolic entities, i.e. character strings and numbers. These may be designated directly in the language, with the intermediate device of a constant name. Character string constants will be shown between double quotes, e.g., "this is a character string", while numeric constants will have the usual Arabic notation, with an optional embedded decimal point, e.g., 1, 4, 5, 98.6. For consistency, these designations will be treated as names for themselves. Thus, the general notation for constants is that they begin with a capital letter, digit, or double quote.

Object Language – Variables Variables will be denoted as one or more lower case letters, with an optional subscript, e.g., x, y, z1, z2.

Meta-Language In the meta-language, constants will be represented using the Greek characters, α, β, γ, Φ, Ψ. Variables will be designated in the meta-language by the characters μ and ν.

[2] [RU71]

2 The Language L_1

L_1 is a fairly standard version of a first order predicate calculus with equality. (See Dowty [Dow78, pages 15–21] for a more detailed discussion.)

2.1 Syntax of L_1

Basic Expressions:

Constants:

Individual constants are denoted as a capital letter followed by one or more lower case letters, e.g., *A, B, Tom, Dick, Harry.*

Propositional and predicate constants are denoted as all capital letters, possibly with embedded hyphens or digits, e.g., *TALL, TALLER-THAN.*

Variables: Individual variables are denoted as one or more lower case letters, e.g., x, y, z. Each predicate has associated zero or more *places.* (A zero place predicate is called a *proposition.* Note: in language L_1 there are no variables for propositions or predicates.)

Terms: A *term* in L_1 is an individual variable or an individual constant.

Formation Rules of L_1: A well formed formula of L_1 is defined recursively as follows:

1. If Φ is a predicate of n places, (n $\geq$ 0), and $\alpha_1, \ldots \alpha_n$ are terms, then $\Phi(\alpha_1, \ldots \alpha_n)$ is a well formed formula;

2-6. If Φ and Ψ are well formed formulae, the so are:
2. $\neg\Phi$
3. $\Phi\&\Psi$
4. $\Phi \vee \Psi$
5. $\Phi \rightarrow \Psi$
6. $\Phi \leftrightarrow \Psi$

7-8. If μ is a variable and Φ a well formed formula, then:
7. $\forall\mu\Phi$ is a well formed formula
8. $\exists\mu\Phi$ is a well formed formula

A variable μ is bound in a formula, Φ, if and only if it occurs in Φ within a sub-formula of the form $\forall\mu\Phi$ or $\exists\mu\Phi$; otherwise the variable is *free* in Φ.

A *sentence* is a well formed formula containing no free variables.

2.2 Semantics of L_1

A model for L_1 is an ordered pair $<D, F>$ such that D (the universe of discourse) is a non-empty set and F (the interpretation function) is a function assigning a denotation to each constant of L_1 (i.e. to individual constants and predicate constants). The set of possible denotations of individual constants is D. The set of possible denotations of one place predicates is ρ(D), where ρ is the powerset of D, i.e. the set of all subsets). the set of possible denotations for an n-place predicate is $\rho(D_n)$, where

$$D_n = \{<d_1, \ldots,d_n>|d_1 \in D, \ldots, d_n \in D\}.$$

The set of possible denotations for a 0-place predicate (proposition) is the set $\{True, False\}$.

An *assignment of values to variables* (or *value assignment*) g is any function assigning a member of D to each variable of L_1. "$Den_{M,g}(\alpha)$" is the abbreviation for "denotation of α with respect to model M and value assignment g"; "true with respect to M, g" abbreviates "true with respect to a model M and value assignment g".

Denotations of Basic Expressions of L_1 (relative to a model $<D, F>$ and value assignment g)

1. If μ is an individual variable of L_1, then $Den_{M,g}$ (μ) = g(μ).

2. If α is a (non-logical) constant of L_1, then $Den_{M,g}(\alpha)$ = F(α).

Truth Conditions for Formulae of L_1 (relative to a model $<D,F>$ and value assignment g)

1. If Φ is an n place predicate and $\alpha_1, \ldots, \alpha_n$ are terms, then $\Phi(\alpha_1, \ldots \alpha_n)$ is true with respect to M, g, if and only if $Den_{m,g} <\alpha_1, \ldots, \alpha_n> \in Den_{M,g}(\Phi)$.

2. If Φ is a well formed formula, then $Den_{M,g}(\neg\Phi) = True$ if and only if $Den_{M,g}(\Phi) = False$

3-6. If Φ and Ψ are well formed formulae, then:

3. $Den_{M,g}(\Phi\&\Psi) = True$ if and only if $Den_{M,g}(\Phi) = True$ and $Den_{M,g}(\Psi) = True$;

4. $Den_{M,g}(\Phi \vee \Psi) = True$ if and only if either $Den_{M,g}(\Phi) = True$ or $Den_{M,g}(\Psi) = True$;

5. $Den_{M,g}(\Phi \rightarrow \Psi) = True$ if and only if either $Den_{M,g}(\Phi) = False$ or $Den_{M,g}(\Psi) = True$;

6. $Den_{M,g}(\Phi \leftrightarrow \Psi) = True$ if and only if either
(a.) $Den_{M,g}(\Phi) = True$ and $Den_{M,g}(\Psi) = True$ or
(b.) $Den_{M,g}(\Phi) = False$ and $Den_{M,g}(\Psi) = False$;

7. If Φ is a formula and μ is a variable, then $Den_{M,g}(\forall\mu\Phi) = True$ if and only if for every value assignment g' such that g' is exactly like g except possibly for the individual assignment of μ by g', $Den_{M,g'}(\Phi) = True$.

8. If Φ is a formula and μ is a variable then $Den_{M,g}(\exists\mu\Phi) = True$ if and only if there is some value assignment, g', such that g' is exactly like g except possibly for the value assigned to μ by g' and $Den_{M,g'}(\Phi) = True$.

Truth Conditions for Formulas of L_1 Relative to a Model M

1. A formula Φ of L_1 is true with respect to M if for all value assignments g, $Den_{M,g}(\Phi) = True$.

2. A formula Φ of L_1 is false with respect to M if for all value assignments g, $Den_{M,g}(\Phi) = False$.

Note: if a formula Φ is a sentence or proposition (i.e. with no free variables), then it will turn out that $Den_{M,g}(\Phi) = True$ with respect to M and all value assignments, g (hence true with respect to M by 1. above) or else $Den_{M,g}(\Phi) = False$ with respect to M and all value assignments (hence false with respect to M by 2. above.) It can never be true with respect to M for some value assignments, and false with respect to M for other value assignments. However, if Φ has one or more free variables, then it may be true with respect to some assignments, and false with respect to others. In this case, its truth or falsity is simply undefined by the above rules.

3 Re-Interpretation of Predicates

In the preceding section, a one place predicate was regarded as denoting a subset of the domain D. Hence, for a term α and a predicate Φ, $\Phi(\alpha)$ is true (denotes $True$) if and only if the thing α denotes is an element of the set denoted by Φ. Similarly for n-place predicates, Φ is viewed as an n-place relation on D, and is true of n terms, $\alpha_1, \ldots, \alpha_n$, if and only if the n-tupe of entities they denote is an element of the relation denoted by Φ. Following Dowty [Dow78, pages 29–30], this interpretation will now be modified slightly. Consider first the case of a one place predicate, P. Suppose we had a domain, D, consisting of five individuals as follows:

$$D = \{D1, D2, D3, D4, D5\}$$

and

$$Den(P) = \{D1, D4, D5\}$$

Here P is true (denotes $True$) of the individuals in the set $\{D1, D4, D5\}$ and is false (denotes $False$) of the individuals not in this set. These denotations of True and False can be made explicit by describing the characteristic function of P. This is a function that maps any individual in D to True or False, according to whether it is in the subset of D denoted by P. For instance, the characteristic function in this case is the set of pairs:

<D1, True>
<D2, False>
<D3, False>
<D4, True>
<D5, True>

The information conveyed here is essentially that of the previous subset plus the interpretation of elementhood conveying the truth of the predicate applied to its argument. However here, this interpretation is conveyed directly. That is, let us henceforth view a one place predicate as denoting the characteristic function of the set of elements for which it is true. Then the denotation of the predicate applied to an argument is simply the result of functional application of this argument to the characteristic function, i.e. if Φ is a one place predicate and α is a term, then

$$Den_{M,g}(\Phi(\alpha)) = Den_{M,g}(\Phi)Den_{M,g}(\alpha).$$

For instance, in the above example, if α is D1, then $Den_{M,g}(\Phi(\alpha)) = True$; if α is D2 then $Den_{M,g}(\Phi(\alpha)) = False$.

We might similarly extend this so that two-place predicates denoted sets of triples, mapping two individuals to a true value, and that n-place predicates denoted n+1 tuples mapping n individuals to a truth value. However, it will provide more flexibility later on if we regard a two place predicate not as a function of two arguments mapping to truth values, but as a function of one argument mapping to another function of one argument that maps to a truth value.

Thus a predicate of any number of places is considered to denote a function of only one argument whose result is either another function or a truth value. (The idea of functions that have other functions as values may seem strange – except perhaps to LISP programmers. Its motivation will become clear when we introduce lambda abstraction.) To incorporate this new interpretation, the language L_1 is modified as follows. Replace formation rule 1 with:

Syn. 1a. If Φ is a one-place predicate and α is a term, then $\Phi(\alpha)$ is a well formed formula.

Syn. 2a. If Φ is an n-place predicate and α is a term, then $\Phi(\alpha)$ is an n-1 place predicate.

Replace semantic rule 1 with:

Sem 1. If Φ is an n-place predicate ($n \geq 1$) and α is a term, then $Den_{M,g}(\Phi(\alpha)) = Den_{M,g}(\Phi)(Den_{M,g}(\alpha))$.

Note that the previous notation $\Phi(\alpha_1, \alpha_2, \ldots, \alpha_n)$ now takes the form

$$\Phi(\alpha_1)(\alpha_2)\ldots(\alpha_n)$$

The former notation will still be used on occasion to abbreviate the latter however.

4 Many-Sorted, Type-Theoretic Languages

A many-sorted formal language is one that assumes there is a non-empty set I whose members are called sorts. For each sort i, there are variables V_{i_1}, V_{i_2}, ... that belong to sort i. Also for each sort i there is a possibly empty set of constant symbols of sort i. For each $n > 0$ and each n-type $<i_1, \ldots, i_n>$ of sorts, there is a (possibly empty) set of predicates, each of which is said to be of sort $<i_1, \ldots, i_n>$. For each sort i there is a universal and existential quantifier, $\forall_i$, and $\exists_i$. A many sorted logic can be embedded in a first order predicate calculus (using special predicates for each sort) and therefore does not have any more power.

A many sorted approach will be useful later when we extend the domain of the formal language to include in addition to entities, character strings, numbers and times. The purpose of a many sorted language is to coordinate references among the several domains of discourse representing each sort. As noted above, these references remain first-order, that is, only individuals and their properties and relationships (indicated by predicates) are represented in the language. Recall that in the previous section we modified the interpretation of a predicate so that it no longer denoted a set or relation on a domain, but rather characteristic functions of such sets. Rather than orient the formal language towards the first order framework of a multi-sorted language, we will instead continue the development begun the last section and introduce a more general framework that includes the multiple domain features of a multi-sorted language. Such a language is called a higher-order type-theoretic language (the name derives from origins in Russell's simple theory of types).

Basically, a type is like a sort as described above, except that a type may be not only a class of individuals (like a sort), but classes of higher order objects (e.g., sets, sets of sets) as well. So far, the elementary types we have discussed are individuals in the domain, designated as type 'e (for "entity" and truth values, which we designate as type 'v (from Latin, *veritas*; the obvious abbreviation t is reserved for time, which appears later).

The set of types, called *Type*, is defined recursively as follows:
(1) e is a type
(2) v is a type
(3) if a and b are any types, then $<a,\ b>$ is a type.

The members of *Type* are labels of categories. The notation ME_a (the meaningful expressions of type a) denotes the set of expressions of type a itself. By way of example:

a formula or proposition is of type v;
a one place predicate is of type $<e,v>$;
a two place predicate is of type $<e,\ <e,v>>$;
negation ($\neg$) is of type $<v,v>$;
connectives (&, V, $\rightarrow, \leftrightarrow$) are all of type $<v,\ <v,\ v>>$.

5 λ Abstraction

Using set notation, a set may be defined extensionally, listing its elements, for example,

$A = \{A1, A2, A3\}$

or "intensionally", by means of some predicate that selects from the domain a subset of individuals, for example,

$$B = \{x|P(x)\}$$

is the set of all individuals satisfying P. This brace notation thus provides the means for constructing higher order sets from a predicate. But, by our first interpretation of predicates, they themselves denoted sets, e.g., Den(P). Thus, substituting,

$$B = \{x|x \in Den(P)\}$$

or,

$$B = Den(P).$$

In our revised interpretation, however, the denotation of P was modified to be the characteristic function of this set. The device for referring to this in the object language is the so-called *lambda operator*, λ. Thus, for P a one place predicate,

$$\lambda x P(x)$$

is the set of ordered pairs of the form $<e,v>$, one pair for each individual in the domain, which assigns $True$ or $False$ if P is true or false of that individual respectively. While we have introduced lambda in terms of individuals and one place predicates, it can in fact be generalized to apply to expressions and variables of any type. This involves the following additions to the syntactic and semantic rules:

Syn. If $\alpha \in ME_a$ and μ is a variable of type b, then $\lambda\mu\alpha \in ME_{<b,a>}$.

Sem. If $\alpha \in ME_a$, and μ is a variable of type b, then $Den_{M,g}(\lambda\mu\alpha)$ is that function h from D_b into D_a such that for all objects k in D_b, h(k) is equal to $Den_{M,g'}(\alpha)$, where g' is that variable assignment exactly like g except for the possible difference that g'(μ) = k.

Note that lambda abstraction takes the role of set definition and functional application takes the role of set membership in the *object languages* we are developing, whereas traditional set concepts are used in the *meta-language* definitions. In later parts, where we illustrate the use of CANDID with examples, it will occasionally be convenient to revert back to traditional set notation because of its familiarity. For this reason, we include the following additional definitions in the *object language*. For a predicate Φ and a

variable μ,

$$\{\mu|\Phi(\mu)\} ::= \lambda\mu[\Phi(\mu)]\mu \in \Phi ::= \Phi(\mu)$$

To repeat, while set notation may thus be used in the *object language*, its interpretation is in terms of lambda abstraction and characteristic functions. In the *meta-language* definitions, set notation is used in the normal way.

6 Operations, Definite Reference

The expressions discussed so far have all been of the type e or $<a,v>$ where 'a' is some other, possibly complex type. An *operation* is an expression of the form $<a,b>$ where b is an elementary type other than v. At the current level of the language, the only expressions that qualify are of the form $<a, e>$, i.e. expressions that result in an individual, when applied to an argument. Indeed, an individual constant may be regarded as a 0-place operation. Operations may thus serve as arguments to predicates. For example, for the predicate *ITALIAN*, the operation, *Father*,

ITALIAN(Father(Roberto))

asserts that Roberto's father is Italian. Note: to aid readability in the examples, we adapt the following practice for constant names: constants denoting individuals (individual constants and operations) are given names beginning with a capital letter, followed by lower case. Other constants, including predicates, are given names all in upper case.

Operations serving as arguments to predicates are included in the definition of functional application given in the preceding section: that is, the argument to a function of type $<a,b>$ may be an meaningful expression of type 'a'. This includes operations as well as variables and constants. For instance, in the above example,

$John \in ME_e$
$Father \in ME_{<e,e>}$
$ITALIAN \in ME_{<e,v>}$

so that, by functional application,

$Father(John) \in ME_e$
$ITALIAN(Father(John)) \in ME_v$

(The quantifiers $\forall$ and $\exists$ as well as the lambda operator, λ, are confined by definition to variables only.) Note that by combining an operation with a

predicate of equality we can define a corresponding predicate:

FATHER(x,y) ::= (y=Father(x))

A new operator, the so-called descriptive or iota operator, IOTA, allows making definitions in the other direction. This operator has the following syntactic and semantic rules:

Syn. If $\Phi \in ME_{<a,v>}$ and μ is a variable of type a, then $\iota\mu\Phi \in ME_a$.

Sem. For $\Phi \in ME_{<a,v>}$ and μ is a variable of type a, if for some constant, c, $Den_{M,g}(\forall\mu[\Phi(\mu) \leftrightarrow \mu = c]) = True$, then $Den_{M,g}(\iota\mu\Phi\mu) = c$.

Note that by this definition, the expression $\iota\mu\Phi\mu$ has a denotation only if Φ is true of just one individual; otherwise, if Φ is true of no individuals or more than one individual, the denotation of $\iota\mu\Phi\mu$ is undefined. Expressions of the form "$\iota\mu\Phi\mu$" are read "the unique μ such that Φ". ι is thus an operation forming operator. For example, the earlier operation *Father(x)*, could be formed from the predicate *FATHER(x,y)* as follows:

λx*Father(x)* ::= $\lambda x \iota y$*FATHER(x,y)*

Comment: By way of comparison, $\iota\mu\Phi\mu$ denotes the set (or rather characteristic function thereof) of individuals satisfying Φ. This may, coincidentally, be a set with only one element (characteristic function with only one domain value mapping to True), or indeed it may be the null set. The expression $\iota\mu\Phi\mu$, on the other hand, denotes a single individual if it denotes at all.

7 Summary of the Language L_v

The language L_v incorporates the features discussed thus far (see also Dowty [Dow78, pages 46–54]).

7.1 Syntax of L_v

The set of types of L_v is the set defined as follows:

(1) e is a type
(2) v is a type
(3) if a and b are any types, then $<a, b>$ is a type.

The basic expressions of L_v consist of:

(1) for each type, a, the set of (non-logical) *constants of type a*, denoted Con_a. Names for particular constants follow the conventions defined earlier: all constant names begin with a capital letter. Names of constants which refer to entities, have the remainder spelled in lower case; all other constants have names spelled entirely in upper case.

(2) for each type, a, the set of *variables of type a*, denoted Var_a (Names for variables are as before; that is, lower case letters with an optional numeric subscript.)

(3) for each type, a, the set of *terms of type a*, denoted $Term_a$, are defined recursively as follows:

— if $\alpha \in Con_a$ then $\alpha \in Term_a$

— if $\alpha \in Var_a$ then $\alpha \in Term_a$

— if $\beta_1, \ldots, \beta_n$ are terms of type $x_1, \ldots, x_n$ respectively and Φ is an operation of type $<x_1, \ldots, <x_n, a>>$ then $\Phi(\beta_1, \ldots, \beta_n) \in Term_a$.

— if $\mu \in Var_a$ and $\Phi \in ME_{<a,t>}$ whose only unbound variable is μ, then $\iota\mu\Phi \in Term_a$.

Formation Rules of L_v The set of meaningful expressions of type, a, denoted ME_a, for any type, a, (the well formed expressions for each type) is defined recursively as follows:

1. For each type, a, every variable and constant of type, a, is in ME_a.
2. If $\alpha \in ME_{<a,b>}$ and $\beta \in ME_a$ then $\alpha(\beta) \in ME_b$.
3. If $\alpha \in ME_a$ and μ is a variable of type, b, then $\lambda\mu\alpha \in ME_{<a,b>}$.
4. If α and β are terms of type, a, then $[\alpha = \beta] \in ME_v$.
5. If $\Phi \in ME_{<a,v>}$ and $\mu \in Var_a$, then $\iota\mu\Phi \in ME_a$.

6-10. If Φ and Ψ are in ME_v, then so are:

6. $[\neg\Phi]$
7. $[\Phi \& \Psi]$
8. $[\Phi \vee \Psi]$
9. $[\Phi \rightarrow \Psi]$
10. $[\Phi \leftrightarrow \Psi]$

11-12. If $\Phi \in ME_v$ and μ is a variable (of any type) then

11. $[\forall\mu\Phi] \in ME_v$
12. $[\exists\mu\Phi] \in ME_v$

7.2 Semantics of L_v

Given a non-empty set D (the domain of *individuals* or *entities*), the set of possible denotation of meaningful expressions of type, a, (abbreviated D_a) is

given by the following recursive definition

(1) $D_e = D$
(2) $D_V = \{True, False\}$
(3) $D_{<a,b>} = D_{b_{D_a}}$ for any types a and b.

(The notation of the form Y_X is the set of all possible functions from the set X into the set Y.) A *model* for L_v is an ordered pair <*D, F*> such that D is as above and F is a function assigning a denotation to each constant of L_v of type, a, from the set D_a. An *assignment of values to variables* (or simply a *variable assignment*) g is a function assigning to each variable $\mu \in Var_a$ denotation from the set D_a, for each type, a. The denotation of an expression α of L_v relative to a model M and variable assignment g is defined recursively as follows:

1. If α is a constant, then $Den_{M,g}(\alpha) = \mathrm{F}(\alpha)$.

2. If α is a variable then $Den_{M,g}(\alpha) = \mathrm{g}(\alpha)$.

3. If $\alpha \in ME_{<a,b>}$ and $\beta \in ME_a$, then
$Den_{M,g}(\alpha(\beta)) = Den_{M,g}(\alpha)(Den_{M,g}(\beta))$.
(That is, the result of applying the function $Den_{M,g}(\alpha)$ to the argument $Den_{M,g}(\beta)$).

4. If $\alpha \in ME_a$ and $\mu \in Var_b$, then $Den_{M,g}(\lambda\mu\alpha)$ is that function, h, from D_b into D_a such that for all objects, k, in D_b, h(k) is equal to $Den_{M,g}$, where g' is that variable assignment exactly like g except for the possible difference that g'(μ) = k.

5. If α and β are terms of type, a, then $Den_{M,g}(\alpha = \beta) = True$ if and only if $Den_{M,g}(\alpha)$ is the same as $Den_{M,g}(\beta)$.

6. For $\Phi = ME_{<a,t>}$ and $\mu \in Var_a$, if for some constant, c, $Den_{M,g}\forall\mu[\Phi\mu \leftrightarrow \mu = c] = True$, then $Den_{M,g}(\iota\mu\Phi\mu) = c$. (Otherwise the expression $\iota\mu\Phi\mu$ has no denotation defined.)

7-11. For Φ and Ψ in ME_v:

7. $Den_{M,g}(\neg\Phi) = True$ if and only if $Den_{M,g}(\Phi) = False$;

8. $Den_{M,g}(\Phi\&\Psi) = True$ if and only if $Den_{M,g}(\Phi) = True$ and $Den_{M,g}(\Psi) = True$;

9. $Den_{M,g}(\Phi \vee \Psi) = True$ if and only if either $Den_{M,g}(\Phi) = True$ or $Den_{M,g}(\Psi) = True$;

10. $Den_{M,g}(\Phi \rightarrow \Psi) = True$ if and only if either $Den_{M,g}(\Phi) = False$ or $Den_{M,g}(\Psi) = True$;

11. $Den_{M,g}(\Phi \leftrightarrow \Psi) = True$ if and only if either (a.) $Den_{M,g}(\Phi) = True$ and $Den_{M,g}(\Psi) = True$ or (b.) $Den_{M,g}(\Phi) = False$ and $Den_{M,g}(\Psi) = False$;

12. If Φ is a formula and μ is a variable, then $Den_{M,g}(\forall \mu \Phi) = True$ if and only if for every value assignment g' such that g' is exactly like g except possibly for the individual assignment of μ by g', $Den_{M,g'}(\Phi) = True$.

13. If Φ is a formula and μ is a variable then $Den_{M,g}(\exists \mu \Phi) = True$ if and only if there is some value assignment, g', such that g' is exactly like g except possibly for the value assigned to μ by g' and $Den'_{M,g}(\Phi) = True$.

8 Character Strings, Labels

We now introduce a new elementary type, called a *character string*, abbreviated by the type name, *c*. The set of types (*Type*) is therefore extended as follows:

e is a type
c is a type
v is a type
if x and y are types, then $<x,y>$ is a type.

The set of elementary characters is the set *Char* where

Char = {*A, B, . . . , Z, 0, 1, . . . , 9, ., -*}

This character set is sufficient for our purposes here. It can be extended as needed to include e.g., lower case letters, special character markings such as accents, circumflex, cedilla, tilde, or completely different alphabets such as Cyrillic or Greek.

The set C of character strings is the set of n-place relations defined on Char, i.e.

$C = \bigcup_{n=1}^{\infty} Char_n$

where $Char_n$ is *Char X Char X Char . . .* n times. A character string constant is therefore an n tuple, $<\alpha_1, \alpha_2, \ldots, \alpha_n>$ where $\alpha_1 \in C$, ..., $\alpha_n \in C$.

This will henceforth be abbreviated,

"$\alpha_1\alpha_2 \ldots \alpha_n$"

i.e. a character string constant is a string of characters from the above set C listed between double quotes.

Various computer languages, such as SNOBOL, provide a rich vocabulary of predicates and operations on strings. Here we make use of only the bare minimum of such a vocabulary, namely predicate of equality which is defined for all types in the calculus. Again this could be extended as needed for diverse applications. Here the principle interest in character strings is with operations of the form <*e,c*>. This is a mapping from an entity to a character string, what we will call a label. Examples of labels are:

Last-Name(x) = "SMITH"
First-Name(y) = "JOHN"
Corp-Name(z) = "GENERAL MOTORS"
Vehicle-Number(a) = "N33E76"
Social-Security-Number(b) = "474-52-4829"

As is probably evident from these examples, a label is an association of a character string with an individual for identification purpose only. Labels may or may not provide unique identification, as the above examples illustrate.

9 Numbers and Measurement

Another elementary type is now added, that of *numbers*, which we take to be the real numbers. The set of numbers is designated as N, and the elementary type, number, is abbreviated n. The set of types is now extended as follows:

e is a type
c is a type
n is a type
v is a type
if x and y are types, then <*x,y*> is a type.

Numeric constants are denoted in the common way as a string of Arabic digits, with an optional imbedded decimal point and an optional leading sign, e.g., 0, 1.2, -3.7. The one-place predicate *INT* (i.e. of type <*n,v*>, designates the set of integers. As for all types, the predicate "=" is assumed. Further, a linear ordering, indicated by the predicate "<" is assumed. Based on these, plus negation, the other numeric inequalities are easily derived. The notation is as follows, for α and β terms of type n:

$\alpha = \beta$ α equals β

$\alpha < \beta$	α less than β
$\alpha \leq \beta$	α less than or equal to β
$\alpha > \beta$	α greater than β
$\alpha = \beta$	α greater than or equal to β
$\alpha \neq \beta$	α not equal to β

These predicates are all of type $<n, <n,v>>$ The following operations, of type $<n, <n,n>>$ are also assumed. For α and β terms of type n:

$\alpha + \beta$	addition
$\alpha - \beta$	subtraction
$\alpha * \beta$	multiplication
α / β	division
$\alpha * * \beta$	exponentiation

Our principle interest in numbers in CANDID is as they are related to entities (and later, times). An operation of type $<x,t>$, where is a term of type e, is called a *measurement function*, that is, it is a mapping from the entities to the numbers (or a subset thereof). For instance,

Height-In-Meters(x) = 6.5

indicates that x is 6.5 meters tall. In the theory of measurement, a measurement is generally taken to include a so-called *measurement operation* and a *measurement standard*. Measurement standards are the sorts of objects maintained by for instance the National Bureau of Standards in Washington, D.C., which have some special property against which other objects are to be gauged. Thus a particular rod is regarded as the standard meter for the country. (A more picturesque example: the roundish stone on the front of St. Stephan's cathedral in the center square of Vienna was used in medieval times as a standard for the size of a load of bread.)

A measurement operation is the procedure by which another object is compared to the standard. This procedure may be direct, e.g., by aligning the object against the standard meter, or indirect, through the use of intermediating measurement devices (rulers, measuring tapes, etc.) which are ultimately compared to the standard.

In the formal language, a measurement operation is regarded as a (formal) operation, which a measurement standard is an individual constant. For instance, we may modify the last example to be:

Height(x, Meter) = 6.5

Here, *Height* is a measurement operation and *Meter* is an individual, which serves as a measurement standard. Note that measurement operations are numeric terms and thus may appear as arguments to other numeric predicates

and operations. For example, to assert measurement unit convertibility from inches to centimeters:

$$\forall x \mathit{Height}(x,\ \mathit{Cm}) = 2.54 * \mathit{Height}(x,\ \mathit{Inch})$$

where *Cm* and *Inch* are measurement standards.

10 Time, Realization, Change

Time Individuals. Another elementary type is now added, consisting of elementary points in *time*. The set of times (past, present, and future) is denoted T, and its corresponding type, t. The set of types is thus extended as follows:

e is a type
c is a type
n is a type
t is a type
v is a type
if x and y are types, then $<x,y>$ is a type.

Equality, "=" and "<", a linear ordering, are assumed as predicates on T. With the aid of negation and disjunction, other temporal relations are defined in a straightforward way. If α and β are terms of type t, these have the following interpretation:

$\alpha = \beta$	α is the same time (point) as β
$\alpha < \beta$	α is earlier than β
$\alpha \leq \beta$	α is earlier than or equal to β
$\alpha > \beta$	α is later than β
$\alpha = \beta$	α is later than or equal to β
$\alpha \neq \beta$	α not equal to β

Lastly, the predicate *NEXT*, indicates adjacent point in time:

$$\mathit{NEXT}(\alpha, \beta) ::= (\alpha < \beta)\& \\ \forall\mu((\mu \neq \alpha)\&(\mu \neq \beta) \rightarrow \neg(\alpha < \mu < \beta))$$

In many cases, our interest is not with points in time, but rather time intervals or *spans*. A time span is the set of points between and including two time points. This is provided by the operation *Span*, of type $<t <t,\ <t,v>>>$:

$$\mathit{Span} ::= \lambda x \lambda y \lambda z[(z \geq x)(z \leq y)]$$

For type time points, α and β,

$Span(\alpha)(\beta)$

is the set (technically, characteristic function) of points between and including these two points. Further, for a third time point, γ,

$Span(\alpha)(\beta)(\gamma)$

evaluates $True$ or $False$ depending whether γ is between α and β or not. (Note: as we have defined it, *Span* can also be used to select an interval of numbers.) Conversely, it is often convenient to go in the opposite direction to obtain the beginning and end points of a time span:

$Beg ::= \lambda x \iota y \exists z[x = Span(y, z)]$
$End ::= \lambda x \exists y \iota z[x = Span(y, z)]$

Thus, for a time span α, $Beg(\alpha)$ is its beginning point; $End(\alpha)$ is its ending point. It is also occasionally useful to express that one time span is contained within another. We call this PT (for part):

$PT ::= \lambda x \lambda y[Beg(x) \geq BEG(y) \& End(x) \leq End(y)]$

Thus for two time spans α and β, $PT(\alpha)$ (β) says that β begins after α begins and ends before α ends. As noted, $Span(\alpha)(\beta)$ results (maps to) a set of time points of type $<t,v>$. Many of these time spans have familiar labels, as provided by the Gregorian calendar, e.g., 28 February, 2001, and 20 July 2004 are two individual day time spans, February, 2001, and July 2004 are two individual month time spans, and 2001 and 2004 are two individual year time spans. Reference to the time span constants labeled by the Gregorian calendar will be provided by three operations:

Date of type $<n, <n, <n <t, v>>>>$
Mo of type $<n, <n <t, v>>>$
Yr of type $<n <t, v>>$

That is, each of these maps (three, two or one) numbers to time spans, where months are specified by an integer 1-12. Thus the operation Date imitates the informal notation, e.g., 28.2.2001. The time span (individuals) mentioned above would thus be referenced as:

Date(28, 2, 2001)
Date(10, 7, 2004)
Mo(2, 2001)
Mo(7, 2004)
Yr(2001)

Yr(2004)

Further, we often want to apply numeric measurement to time spans. For this we use the measurement operation *Dur* (for duration). Thus for a time span α, a temporal measurement standard β and a number γ,

$$Dur(\alpha, \beta) = \gamma$$

is read that the duration of α in terms of β is γ. The choice of measurement standards is however somewhat problematic in the case of time spans. Standards such as *Second*, *Minute* and *Hour* pose no particular problems since these are precisely determined based on a particular physical phenomenon (e.g., movement of a standard pendulum, molecular vibrations of quartz crystals). Generally for commercial purposes however, we have need of larger size units, e.g., days, months, years. Following the procedure recommended earlier, suppose we chose one particular month to serve as our standard, say January, 2001. Then, the duration of year, e.g., 2003, in terms of this standard month would be:

Dur(Yr(2003), Mo(1, 2001)) = 11.77.

If, however, we take the next month as our standard, February, 2001, we would have:

Dur(Yr(2003), Mo(2, 2001)) = 13.04.

Neither of these accords with the popular usage that a year comprises twelve months. A similar, though slightly less serious problem arises in the choice of a standard year, since leap years do not have the same number of days as other years. Indeed, even the choice of a standard day has potential difficulties, since the length of the last day in a century is slightly longer than the rest. This however seems to be a tolerable level of inaccuracy. Thus, we may take as our standard, call it Day, any of the non-end-of-the-century days or equivalently, define it in terms of standard hours, minutes or seconds. Thus, for example,

Dur(Mo(1, 2001), Day) = 31.
Dur(Mo(2, 2001), Day) = 28.
Dur(Date(1,1, 2001), Day) = 1.

Temporal Realization We next consider the association of times to entities. For this we adopt a notation suggested by Rescher and Urquhart[3] where for

[3] [RU71]

a time point, α, and a formula Φ,

$$R(\alpha)\Phi$$

is read that Φ is "realized" at time α. That is, if Φ is the formula "it is raining", this expression would be true at certain times, false at others. Including this in the formal language would obviously require a syntactic rule like:

Syn. If α is a term of type t, and $\Phi \in ME_v$, then $R(\alpha)\Phi$ is in ME_v.

However, the inclusion of the R operator will lead us to revise our semantic format somewhat. Like character strings and numbers, time points are merely another sort added to the object language. Viewed this way, the R operator is simply a functional application, i.e.,

$$R(\alpha)\Phi ::= \Phi(\alpha)$$

(This would of course assume that a variable ranging over time points was lambda abstracted within Φ.) However, in order to make various needed discriminations in the semantic rules, we prefer to take a different tack: in additional to including time in the object language, we will also include it in the meta-language. That is to say, time is not only another sort or type within the object language, but will also figure as an additional dimension on which the denotation depends in the meta-language. Or, one may regard it as though there were actually two times involved: those referred to within the expression, and the time of the expression itself. In order to make the separation clear, we will use variables beginning with "t" in the object language to stand for times. In the meta-language we will indicate times as "j". (This latter maintains a notational convention begun by Montague.) Thus, where we former wrote $Den_{M,g}\Phi$, we will now write $Den_{M,j,g}\Phi$, The semantic rule for R is therefore as follows:

Sem. For α a term of type t, and $\Phi \in ME_v$, $Den_{M,j,g}R(\alpha)\Phi = True$ if and only if for some j' , $\alpha = j'$ and $j' < j$, $Den_{M,j',g}\Phi = True$.

Some explanation might be in order. Here, and henceforth, j will be the time when the expression in question is interpreted, i.e. when the denotation is evaluated (in computer terms, the time when the database is queried). $R(\alpha)\Phi$ is true at this time if and only if Φ is true at some earlier time, α. Note that if α refers to some *future* time, i.e. $\alpha > j$, then the denotation of the expression $R(\alpha)\Phi$ remains undefined by this semantic rule. Several further realization operators will prove useful. They are defined as follows. For a time span α, and a formula Φ:

$$RT(\alpha)\Phi ::= \forall t\alpha(t) \rightarrow R(t)\Phi$$

Reading: Φ is "realized throughout" time span α. Note: since time spans were defined as characteristic functions, the expression "α(t)" evaluates $True$ if time point t is in α.

$$RD(\alpha)\Phi ::= \exists\mu[PT(\mu,\alpha)\&RT(\mu)\Phi]$$

Reading: Φ is "realized during" time span α, i.e. it is realized throughout some sub-interval of α. For a time point, γ, and a formula Φ,

$$RB(\gamma)\Phi ::= \exists\mu[(\mu <= \gamma)\&(R(\mu)\Phi)]$$

Reading: Φ is "realized before" time γ; there is some other time μ earlier than γ when Φ is realized.

Change. The above realization operators are "state oriented", i.e. they indicate something to be true at a particular point or span of time. Another construct will allow us to describe *change*. One could describe change using the above constructs, e.g.,

$$\exists t_0\exists t_1 NEXT(t_0,t_1)\&R(t_0)\Phi\&R(t_1)\Psi$$

where t_0 and t_1 are succeeding moments in time. However, often we will want to describe changes generically, without reference to the specific time when it is occurred. For this we adopt a notation of Von Wright [VW65], where

$$(\alpha\ T\ \beta)$$

is read "α and then β". Here in CANDID, this will be defined essentially as a lambda abstraction on the preceding formula:

$$(\alpha\ T\ \beta) ::= \lambda t_0\exists t_1 NEXT(t_0,t_1)\&\alpha(T_0)\&\beta(t_1).$$

It will be remembered that the set D was defined as consisting of physical objects existing in the past or present. However, it is often necessary to indicate just *when* a particular object exists. For that we need to adopt the predicate,

$$EXISTS(\mu)$$

With the aid of the preceding realization operators, we can indicate whether an object existed at a particular time, e.g.

$RT(YR(2000))EXISTS(John)$

indicates that John existed throughout the year 2000 (he may have existed at other times as well). Birth and death are designated respectively as

$BIRTH(x) ::= \neg EXISTS(x)\ T\ EXISTS(x)$
$DEATH(x) ::= EXISTS(x)\ T\ \neg EXISTS(x)$

One may question how this differs from the existential quantifier, $\exists$, which is sometimes read as "there exists". Rescher and Urquhart[4] offer the interpretation that the predicate, $\exists$, is one of "temporal existence". In our case this is merely a question of convenient readings of the two symbols. The existential quantifier refers to the inclusion of some individual in the model (domain of individuals). The existence predicate EXISTS, however, refers to the relationship between this individual and points or spans of time.

11 Possible Worlds, Intensions

In the last section, we generalized the notion of denotation to depend not only on the model $M = <D, C, N, T, F>$ and an assignment of values to variables, g, but also on the location of the expression in a time dimension. In this section we generalize one final time on the notion of denotation, making it in addition dependent on its location in a so-called *possible world*. This concept has had a rich and not uncontroversial recent history in logic, philosophy and linguistics. The early Wittgenstein [Wit21] saw this as the key to the formalization of natural languages. Later in life, after an immense following was pursuing his earlier work, he reversed this claim [Wit58]. Kripke [Kri63] used the concept of possible world to create a formal semantics for modal logic. On the one hand, mathematical logicians, e.g., Chang and Keisler [CK73], Kalish et al. [KMM80], equate the notion with a model for a formal language (at least at the level of first order languages). On the other hand, linguists and philosophers, e.g., Cresswell [Cre73] and Rescher [Res75], seem to regard possible worlds more broadly, as a sort of gedanken experiment, not limited by the vocabulary of the language.

Our usage of possible worlds here will be more on the mathematical side, i.e. that a possible world is an alternative model. Following Montague's notation, the collection of possible worlds will be designated by the set W, whose individuals are written as i, i', etc. in the meta-language. Apart from the model M and assignment g, the denotation of an expression therefore depends on its location in a possible world, i, and a time, j. The pair, $<i, j>$, is called an *index*.

In the previous formal summary, of the language L_t, the model consisted of the domain, D, of individual entities, and F an interpretation function on D

[4] [RU71]

interpreting the predicate and operation constants as relations and functions on D. Since then, we added the additional sets C (Character strings), N (numbers) and T (times) to the model. Our use of character strings and numbers was essentially an alternative to introducing more predicate names, e.g., *Height(x) = 20* might be viewed as an abbreviation of *HEIGHT-IS-20(x)* and *Last-Name(x) = "SMITH"* might abbreviate *LAST-NAME-SMITH(x)*.

Time, on the other hand, introduced a dimension on which the truth value denotations of an expression depended. That is, for an expression Φ, $Den_{M,j,g}\Phi = True$ or $False$ depending, *inter alia*, on the time j. Here $M =$ <*D, C, N, T, F*>. The only one of these sets that varies with time is D, that is, the set of individuals existing at or before time j. Correspondingly, the interpretation function, F, will also depend on the time, j, since while F includes relations in all the sets, the relations involved in D will vary. Thus, it is essentially only the pair <*D,F*> that vary with j. Here the changes in <*D,F*> as j increases might be viewed as all "due to natural causes", that is, individuals are born and dies; and single individuals change their properties.

The aspects of the model that vary between different possible worlds are also confined to the pair <*D,F*>. Here, however, the differences in <*D, F*> between one possible world and another are arbitrary. (There is no notion of adjacency between possible worlds as there is with times, since worlds are not ordered under "<", hence there is no basis for graduating differences.) Indeed, while we will continue to discuss the pair <*D,F*> as depending on a possible world i, though in an arbitrary way, in fact a possible world is *equivalent* to some arbitrarily chosen domain and interpretation function, that is, some <*D', F'*>. Thus, possible worlds and points in time determine a coordinate system on which <*D,F*> depends. Graphically, we might represent this for two possible worlds, i_1, and i_2, and three times, j_1, j_2, and j_3, as shown in Figure 1.

The purpose, to Montague, of this device is to explicitly represent what philosophers call the *intension* (spelled with an "s") of an expression. (Thus Montague's calculus is called "Intensional Logic"). Very briefly, it has long been recognized that the usual concept of denotation is insufficient to capture what we consider its complete meaning. (In informal usage, this residual part of meaning is often called its *connotation*. Intension and extension, as used here, are more technical terms corresponding to connotation and denotation, respectively.) Frege [Fre93] captured the problem succinctly in his famous example of Morning Star and Evening Star: the two phrases denote the same thing, but they have somewhat different uses, hence different connotations or intensions. More to the point of our interests is the problem of so called *opaque contexts*. In English, these appear with such verbs as "believe", "think", "imagine", etc. followed by the relative pronoun "that". (In Latin based languages these are the class of subjunctive constructions.) Consider the following example. Let

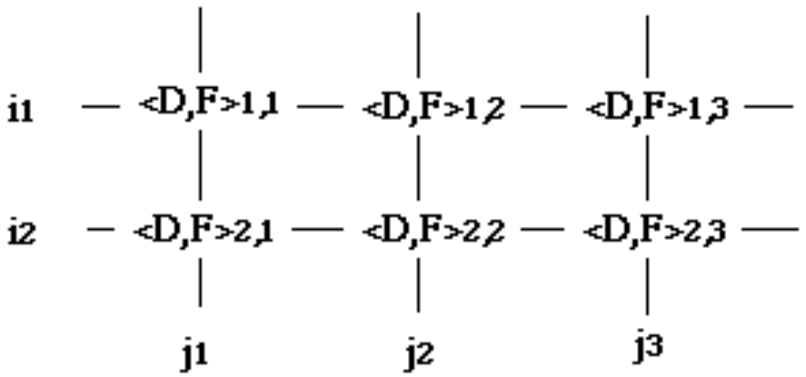


Fig. 1. Coordinate System of Possible Worlds and Times

P = "the world is flat"

Q = "the moon is made of green cheese"

Suppose that an individual, John, believes P, i.e.

BELIEVES(John, P).

The problem is that this would lead us to infer:

BELIEVES(John, Q).

Since P and Q denote the same thing, namely *False*. However, we intuitively find it unacceptable to infer that if someone believes one false thing, he/she then believes every false thing. As relates to the applications of CANDID, this same problem of opaque contexts arises in all types of commercial and financial contracts: if someone contracts to do *some* thing, we do not want to infer that they are obliged to do *every* thing. The mechanism that Montague proposes to avoid this is his *intension operator*, $\bigwedge$. Effectively, this operates as an implicit lambda abstraction on indices (possible world, point-in-time pairs). For instance, $\Phi \in ME_v$, $\bigwedge \Phi$ refers to the set of tuples of the form $<<i,j>, v>$, i.e. evaluating the proposition Φ at every index. Cresswell [Cre73, pages 23–4] offers an intuitive motivation of what this provides:

> If we think for a moment of the job a proposition has to do we see that it must be something which can be true or false, not only in the actual world, but in each possible world. Suppose for the moment that we could "show" a person all possible worlds in turn. This is of

> course impossible, but try to imagine it anyway. We want to know whether two people are thinking of the same proposition. So we ask them, as we show them each (complete) possible world. "Would the proposition you are think of be true if that was the way things were?" If their answers agree for every possible world, there is at least the temptation to suppose that they have the same proposition in mind. Or to put it another way, if the set of worlds to which A says "yes" is the same as the set of worlds to which B says "yes" we can say that A and B have the same proposition in mind. So why not simply identify the proposition with the set of worlds in question? As a first approximation, therefore, we shall say that a proposition is a set of possible worlds.

Thus, we reference to our previous example, we would avoid the erroneous substitution by writing:

$$BELIEVES(John, \bigwedge P).$$

Since there are conceivable possible worlds in which P is true and Q false, or vice versa, $\bigwedge P \neq \bigwedge Q$, even though both P and Q are false in the actual world. The converse of the intension operator is written "$\bigvee$"; that is, $\bigvee \Phi$ is the application of the intension Φ to the actual world. Hence,

$$\bigvee[\bigwedge \Phi] = \Phi.$$

This latter notation will however be of lesser importance for our applications.

As seen in the above discussion, intension and extension are inter-related concepts. Further, extension corresponds to what we have heretofore called denotation. In keeping with the terminology of Montague (and Dowty), we will switch to the abbreviation "Ext" (for extension) rather than "Den" in the semantic rules. Correspondingly, the new abbreviation, "In" (for intension) will be introduced. Let us now summarize the formal language as it stands thus far.

12 Summary of the Language IL

Corresponding to each type, a, the intension of that type will be a new type, written <*s, a*> (where s stands for "sense" – Frege's original term for "intension", which was introduced by Carnap.) The "s" may be read as an abbreviation for the <*i,j*>. Hence <*s, a*> abbreviates <<*i,j*>*, a*>. The set of types (i.e. *Type*) is defined recursively as follows:

v is a type
e is a type

c is a type
n is a type
t is a type
If a and b are types, then $<a,b>$ is a type.
If a is any type, then $<s,a>$ is a type.

The basic expressions of IL consist of:

1) For each type a, the set of *constants of type a* denoted Con_a. (Names for particular constants follow the conventions described earlier.)

2) For each type, a, the set of *variables of type a*, denoted Var_a. (Names for variables follow the conventions described earlier.)

3) For each type a, the set of *terms of type a* denoted $Term_a$, defined recursively as follows:

— if $\alpha \in Con_a$ then $\alpha \in Term_a$
— if $\alpha \in Var_a$ then $\alpha \in Term_a$
— if $\beta_1, \ldots, \beta_n$ are terms of types $x_1, \ldots, x_n$
respectively and Φ is an operation of type
$<x_1, \ldots, <x_n, a>>$
then $\Phi(\beta_1, \ldots, \beta_n) \in Term_a$.
— if $\mu \in Var_a$ and $\Phi \in ME_{<a,t>}$, whose only unbound variable is μ, then $\iota\mu\Phi \in Term_a$.

The set of *meaningful expressions* of type a, denoted ME_a, is defined recursively as follows:

Syn_{IL} 1: Every term of type a is in ME_a
Syn_{IL} 2: If $\alpha \in ME_{<a,b>}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.
Syn_{IL} 3: If $\alpha \in ME_a$ and μ is a variable of type b, then $\lambda\mu\alpha \in ME_{<b,a>}$
Syn_{IL} 4: If α and β are both in ME_a, then $\alpha = \beta \in ME_v$.
Syn_{IL} 5: If $\Phi \in ME_v$ and $\mu \in Var_a$ then $\iota\mu\Phi \in ME_a$

Syn_{IL}6-10: If Φ and Ψ are in ME_v, then so are:
Syn_{IL} 6: $\neg\Phi$
Syn_{IL} 7: $\Phi \;\&\; \Psi$
Syn_{IL} 8: $\Phi \vee \Psi$
Syn_{IL} 9: $\Phi \rightarrow \Psi$
Syn_{IL} 10: $\Phi \leftrightarrow \Psi$

Syn_{IL} 11-12: If $\Phi \in ME_v$ and μ is a variable of any type, then
Syn_{IL} 11: $\forall\mu\Phi \in ME_v$

Syn_{IL} 12: $\exists\mu\Phi \in ME_v$
Syn_{IL} 13: if $\mu \in Var_t$ and $\Phi \in ME_v$ then $R\mu\Phi \in ME_v$
Syn_{IL} 14: If Φ and Ψ are in ME_v, then $\Phi\ T\ \Psi \in ME_v$
Syn_{IL} 15: If $\Phi \in ME_v$, then $\bigwedge[\Phi] \in ME_{<s,v>}$
Syn_{IL} 16: If $\Phi \in ME_{<s,v>}$, then $\bigvee \Phi \in ME_v$

12.1 Semantics of IL

A *model* for IL is an ordered tuple $M = <D,\ C,\ N,\ T,\ W,\ F>$ where D, C, N, T and W are non-empty sets, that assign an appropriate denotation to each (non-logical) constant of IL relative to each pair <i,j>, for i ∈ W and j ∈ T. Thus "$F(<i, j>, \alpha) = \gamma$" asserts that the denotation of the constant α in the possible world i at time j is the object γ.)) The set of *possible denotations* of type a, written D_a, is defined as follows (where a and b are any types):

$D_e = D$
$D_c = C$
$D_n = N$
$D_t = T$
$D_v = \{True, False\}$
$D_{<a,b>} = D_{b_{D_a}}$
$D_{<s,a>} = D_{a_{WxT}}$

(where WxT is the set of all world, time point pairs, i.e. the set of all indices <i,j>).

12.2 Semantic Rules

Sem_{IL} 1: If α is a non-logical constant, then $Ext_{M,i,j,g}(\alpha) = F(\alpha)(¡i,j¿)$. (That is, the extension of α at $<i,\ j>$ is simply the result of applying the intension of α, which is supplied by F, to $<i,\ j>$).

Sem_{IL} 2: If α is a variable, then $Ext_{M,i,j,g}(\alpha) = g(\alpha)$.

Sem_{IL} 3: If $\alpha \in ME<a, b>$ and μ is a variable of type b, then $Ext_{M,i,j,g}$ $(\lambda\mu\alpha)$ is that function h with domain D_b such that for any object x in that domain, h(x) = $Ext_{M,i,j,g'}(\alpha)$, where g' is that value assignment exactly like g with the possible difference that $g'(u)$ is the object x.

Sem_{IL} 4: If $\alpha \in Ext_{<a,b>}$ and $\beta \in ME_a$, then $Ext_{M,i,j,g}(\alpha(\beta))$ is $Ext_{M,i,j,g}(\alpha)(Ext_{M,i,j,g}(\beta))$ (i.e., the result of applying the function $Ext_{M,i,j,g}(\alpha)$ to the argument $Ext_{M,i,j,g}(\beta)$).

Sem_{IL} 5: If α and β are in ME_a, then $Ext_{M,i,j,g}(\alpha = \beta)$ is $True$ if and only if $Ext_{M,i,j,g}(\alpha)$ is the same as $Ext_{M,i,j,g}(\beta)$.

Sem_{IL} 6: If $\Phi \in ME_v$, then $Ext_{M,i,j,g}(\neg\Phi)$ is $True$ if and only if $Ext_{M,i,j,g}(\Phi)$ is $False$, and $Ext_{M,i,j,g}(\neg\Phi)$ is $False$ otherwise.

Sem_{IL} 7: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi\&\Psi]$ is $True$ if and only if both $Ext_{M,i,j,g}(\Phi)$ and $Ext_{M,i,j,g}(\Psi)$ are True.

Sem_{IL} 8: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \vee \Psi]$ is $True$ if and only if either $Ext_{M,i,j,g}(\Phi)$ is $True$ or $Ext_{M,i,j,g}(\Psi)$ is $True$.

Sem_{IL} 9: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \rightarrow \Psi]$ is $True$ if and only if either $Ext_{M,i,j,g}(\Phi)$ is $False$ or $Ext_{M,i,j,g}(\Psi)$ is $True$.

Sem_{IL} 10: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \leftrightarrow \Psi] = True$ if and only if either both $Ext_{M,i,j,g}(\Phi)$ and $Ext_{M,i,j,g}(\Psi)$ are both $True$ or both $False$.

Sem_{IL} 11: If $\Phi \in ME_v$ and μ is a variable of type e, then $Ext_{M,i,j,g}(\forall\mu\Phi) = True$ if and only if $Ext_{M,i,j,g'}(\Phi) = True$ for all value assignments g' exactly like g except possibly for the value assigned to μ.

Sem_{IL} 12: If $\Phi \in ME_v$ and μ is a variable of type e, then $Ext_{M,i,j,g}(\exists\mu\Phi) = True$ if and only if $Ext_{M,i,j,g}(\Phi) = True$ for some value assignment g' exactly like g except possibly for the value assigned to μ.

Sem_{IL} 13: For α a term of type t, and $\Phi \in ME_v$, then $Ext_{M,i,j,g}[R\alpha\Phi] = True$ if and only if for time $j' = F(\alpha)$, and $j' < j$, $Ext_{M,i,j',g}[\Phi] = True$.

Sem_{IL} 14:. If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}(\Phi\ T\ \Psi)$ is $True$ if and only if $Ext_{M,i,j,g}(\Phi)$ is $True$ and $Ext_{M,i,j',g}(\Psi)$ is $True$ for the unique j' such that $j < j'$ and for all j'', either not $j < j'' < j'$ or $j'' = j'$.

Sem_{IL} 15: If $\alpha \in ME_\alpha$, then $Ext_{M,i,j,g}(\bigwedge \alpha)$ is that function h with domain WxT such that for all <i', j'> in WxT, $h(<i', j'>)$ is $Ext_{M,i',j',g}(\alpha)$.

Sem_{IL} 16: If $\alpha \in ME_{s,a}$, then $Ext_{M,i,j,g}(\bigvee \alpha)$ is $Ext_{M,i,j,g}(\alpha) = (<i,j>)$ (i.e., the result of applying the function $Ext_{M,i,j,g}(\alpha)$, to the argument <i, j>).

Additional Primitive and Derived Predicates and Operations

For Domain C (character strings) None

For Domain N (numbers)

1. type: $<n, <n, v>>$
   $[\alpha < \beta]$ primitive
   $[\alpha \leq \beta] ::= [\alpha < \beta] \vee [\alpha = \beta]$
   $[\alpha > \beta] ::= \neg[\alpha \leq \beta]$
   $[\alpha \geq \beta] ::= [\alpha > \beta] \vee [\alpha = \beta]$
   $[\alpha \neq \beta] ::= \neg[\alpha = \beta]$

2. type: $<n, <n, n>>$
   $[\alpha + \beta]$ primitive
   $[\alpha - \beta]$ primitive
   $[\alpha / \beta]$ primitive
   $[\alpha * \beta]$ primitive
   $[\alpha * * \beta]$ primitive

For Domain T (times)

1. type: $<t, <t, v>>$
   $[\alpha < \beta]$ primitive
   $[\alpha \leq \beta] ::= [\alpha < \beta] \vee [\alpha = \beta]$
   $[\alpha > \beta] ::= \neg[\alpha \leq \beta]$
   $[\alpha \geq \beta] ::= [\alpha > \beta] \vee [\alpha = \beta]$
   $[\alpha \neq \beta] ::= \neg[\alpha = \beta]$
   $NEXT(\alpha, \beta) ::= (\alpha < \beta) \& \forall t[(t \neq \alpha) \& (t \neq \beta) \rightarrow \neg[\alpha < t < \beta]]$
2. type: $<t, <t, <t, v>>>$
   $Span ::= \lambda x \lambda y \lambda z[z \geq x \& z \leq y]$
3. type: $<<t, v>, t>$
   $Beg ::= \lambda x \iota y \exists z[x = Span(y, z)]$
   $End ::= \lambda x \iota z \exists y[x = Span(y, z)]$
4. type: $<<t, v>, v>$
   $PT ::= \lambda x \lambda y[(Beg(x) \geq Beg(y)) \& (End(x) \leq End(y))]$
5. type: $<n, <n, <n, <, v>>>>$
   $Date(\alpha, \beta, \gamma)$ (primitive)
6. type: $<n, <n, <t, v>>>$
   $Mo(\alpha, \beta)$ (primitive)
7. type: $<n, <t, v>>$
   $Yr(\alpha)$ (primitive)
8. type: $<<t, v>, n>$
   $Dur(\alpha, \beta)$ (primitive)
9. type: t
   Day (primitive, measurement standard individual)

10. type: $<<t,\ v>,\ <v,\ v>>$
    $RT(\alpha)\Phi ::= \forall t(t \in \alpha) \rightarrow R(t)\Phi$
    $RD(\alpha)\Phi ::= \exists\mu(PT(\mu, \alpha)\&R(t)\Phi$
11. type: $<t,<v,\ v>>$
    $RB(\gamma)\Phi ::= \exists\mu[(\mu \leq \gamma)\&RD(Span(\gamma, \mu))\Phi]$

13 Action

Earlier, the connective T, a construct due to Von Wright [VW65], was introduced in order to describe generic changes. We now follow Von Wright's development further to obtain a description of *actions*. Von Wright introduces another connective, I, with a syntax like that of T:

Syn. If Φ and Ψ are in ME_v, then $\Phi\ I\ \Psi \in ME_v$.

This connective has the reading "instead of". Its effect is that, due to the intercession of some agent, Φ is true instead of Ψ being true. As Von Wright points out, I serves to coordinate two possible worlds. Interpreting Von Wright's sense for I in the Montague framework developed it so far, we have:

Sem. For Φ and Ψ in ME_v, then $Ext_{M,i,j,g}(\Phi\ I\ \Psi) = True$ if and only if $Ext_{m,i,j,g}(\Phi) = True$ and $Ext_{m,i',j,g}(\Psi) = True$ for some i' just like i except that i' lacks the interference of some agent.

We extend Von Wright's notion slightly by adding a place in the connective I which specifies the agent. Thus,

Syn. If Φ and Ψ are in ME_v, and α is a term of type e, then $[\Phi(\ I\alpha)\Psi] \in ME_v$.

The corresponding semantic rule is as follows:

Sem'. For Φ and Ψ in ME_v , and α is a term of type e, then
$Ext_{M,i,j,g}(\Phi(\ I\alpha)\Psi) = True$ if and only if
$Ext_{m,i,j,g}(\Phi) = True$ and $Ext_{m,i',j,g}(\Psi) = True$
for some i' just like i except that i' lacks the interference of agent α.

This concept of "interference" is admittedly rather uncomfortable. If we compare the models <D,F> and <D', F'> of i and i' respectively, what is different about them? Precisely that Φ is true in the first, and Ψ is true in the second. This *is* the interference. The I connective combines with T to form what Von Wright calls "TI expressions". It is these expressions that are used to express actions; that is,

$\alpha\ T\ (\beta\ I\ \mu\gamma)$

is read: α is true and then β is true instead of γ due to the interference of μ. For instance, if the action is for John to open a window, we would have:

CLOSED(Window) T (OPEN(Window) I(John) CLOSED(Window))

i.e., the window was closed and then it was open instead of remaining closed due to the interference of John.

14 Modals, Deontic Operators

Montague's Intensional Logic includes the modal operators $\Diamond$ and $\Box$, for possibility and necessity, respectively, by means of the following syntactic and semantic rules:

Syn. 1. If $Phi \in ME_V$ the $\Diamond\Phi \in ME_v$.

Syn. 2. If $Phi \in ME_V$ then $\Box\Phi \in ME_v$.

Sem. 1. For $Phi \in ME_v$
$Ext_{M,i,j,g}(\Diamond\Phi) = True$ if and only if
$Ext_{M,i',j',g}(Phi) = True$ for some i' in W and some j' in T.

Sem. 2. For $Phi \in ME_v$
$Ext_{M,i,j,g}(\Box\Phi) = True$ if and only if
$Ext_{M,i',j',g}(Phi) = True$ for all i' in W and all j' in T.

Thus, $\Diamond\Phi$ indicates that Φ is possibly true, i.e. that it is true in some possible world at some time. Correspondingly, $\Box\Phi$ indicates that Φ is necessarily true, i.e. true in all possible worlds at all times. Either one of these rules could have been omitted, recognizing that the two concepts are inter-definable as logical duals:

$\Box\Phi ::= \neg\Diamond\neg\Phi$

or

$\Diamond\Phi ::= \neg\Box\neg\Phi$

(This follows from the inter-definability of the quantifiers $\forall$ and $\exists$, implicit in the semantic interpretation of these operators.) Von Wright points out that this is only one version of necessity (or possibility), when he calls logical

necessity (or logical possibility). If $\Box\Phi$ then Φ is true by virtue of the interaction of the truth assignments of its composite formulae, independent of what these formulae denote – i.e. Φ is true in all possible worlds. Alternative terminology is that Φ is a *tautology* or that Φ is analytically true.

Another version of necessity (or, possibility) is what Von Wright calls *natural* necessity (possibility). We write this as

$$\Box_N\Phi$$

and

$$\Diamond_N\Phi$$

If $\Box_N\Phi$ is true the Φ is true in all worlds and at all times "because of the way the world operates". Natural necessity is stronger than logical necessity. For instance, "if x is a human, then x is warm blooded" is a natural necessity, though not a logical one. In order to portray natural necessity or possibility in the semantic framework, we would need to qualify certain possible worlds as being "natural", i.e. conforming to the laws of nature. Call this the set W_N such that $W_N \subseteq W$. The syntactic and semantic rules would therefore be as follows:

Syn'1. If $\Phi \in ME_v$ then $\Diamond_N\Phi \in ME_v$.

Syn'2. If $\Phi \in ME_v$ then $\Box_N\Phi \in ME_v$.

Sem'1. If $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Diamond_N\Phi) = True$ if and only if $Ext_{M,i',j',g}(\Diamond_N\Phi) = True$ for some $i' \in W_N$ and some $j' \in T$

Sem'2. If $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Box_N\Phi) = True$ if and only if $Ext_{M,i',j',g}(\Box_N\Phi) = True$ for all $i' \in W_N$ and all $j' \in T$

The logical duality of these concepts again holds, i.e.

$$\Box_N\Phi ::= \neg\Diamond_N\neg\Phi \text{ or } \qquad \Diamond_N\Phi ::= \neg\Box_N\neg\Phi$$

Von Wright extends this one step further to address the concepts of permission and obligation, which he calls the *deontic* modalities. We will abbreviate these as:

$\Diamond_D\Phi$ for Phi is permitted, and

$\Box_N\Phi$ for Phi is obligatory.

(Von Wright uses the notation P and O here, but we reserve that for later uses.) To describe this in the semantic framework, we need to further qualify certain possible worlds as being legitimate within a general ethical or legal code. (Given that there are numerous such codes, e.g., for different countries, there are correspondingly different definitions of permission and obligation. We ignore this aspect for present purposes.) Let us denote the set of deontically permissible worlds as W_D, where,

$$W_D \subseteq W$$

The syntactic and semantic rules are of similar form as before:

Syn"1. If $\Phi \in ME_v$ then $\Diamond_D \Phi \in ME_v$.

Syn"2. If $\Phi \in ME_v$ then $\Box_D \Phi \in ME_v$.

Sem"1. If $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Diamond_D \Phi) = True$ if and only if $Ext_{M,i',j',g}(\Diamond_D \Phi) = True$ for some $i' \in W_N$ and some $j' \in T$

Sem"2. If $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Box_D \Phi) = True$ if and only if $Ext_{M,i',j',g}(\Box_D \Phi) = True$ for all $i' \in W_N$ and all $j' \in T$

Once again, these are logical duals:

$\Box_D \Phi ::= \neg \Diamond_D \neg \Phi$

or

$\Diamond_D \Phi ::= \neg \Box_D \neg \Phi$

That is, if something is obligatory, it is not permissible not to do it. Contrariwise, if something is permitted, is is not obligatory not to do it. Prohibition, i.e. that something is forbidden, is a deontic impossibility, i.e. the negation of permissibility:

$$\neg \Diamond_D \Phi$$

says it is not permitted (forbidden) to do Φ. It is often argued that "ought" implies "can" – i.e. that if something is obligatory then it should be naturally possible. This would be reflected in the assumption:

$$W_D \subseteq W_N \subseteq W$$

The deontic modalities differ from the other modalities in that they generally apply only to *actions*. In the Von Wright representation for actions (there may be others), this suggests that Φ be a TI expression. We would therefore write,

$\Diamond_D(\alpha\ T\ (\beta I \mu \gamma))$

to indicate that μ is permitted to bring about β from the previous state α, instead of allowing γ to occur, and

$\Box_D(\alpha\ T\ (\beta I \mu \gamma))$

to indicate that μ is obliged to bring about β from the previous state α, instead of allowing γ to occur.

14.1 Note on Contingent Obligations, Permissions

A contingent obligation (or permission) is where an action Φ is obligatory (permitted) if another action, Ψ occurs. Considering first the case of contingent obligation, there seems to be two possible representations:

a) $\Box_D[\Psi \rightarrow \Phi]$ (it is obligatory that Psi implies Φ)

b) $\Psi \rightarrow \Box_D[\Phi]$ (if Psi implies it is obligatory to Φ)

The English reading in these two cases does little to help choose between them – both readings seem adequate. However, if we examine the semantic interpretations in both cases there is an important difference. We have

a) $Sem_a\ ExtM, i, j, g\Box_D[\Psi \rightarrow \Phi] = True$ if and only if
$ExtM, i,' j', g\Box_D[\Psi \rightarrow \Phi] = True$ for all $i' \in W_D$ and $j' \in T$

b) $Sem_b\ \Psi \rightarrow \Box_D[\Phi]$ if and only if
$ExtM, i,' j', g(\Psi) = False$ or $ExtM, i,' j', g(\Phi) = True$ for all $i' \in W_D$ and $j' \in T$

In case a, $\Psi \rightarrow \Phi$ must be true at all indices involving permissible worlds (i.e. elements fo W_D). In case b, if Ψ is true at the current index, then Φ must be true at all indices involving permissible worlds. The point is that in the second case, Ψ and Φ do not necessarily apply to the same possible world. Thus, if Ψ were not true at the current index, but were true at some other index involving a permissible world, Φ would not necessarily hold at this other index. This problem is however avoided in case a, and is thus the preferred method of representing contingent obligation. Analogous arguments hold in the case of contingent permission.

14.2 Contractual Obligation and Permission

The concepts of obligation and permission discussed thus far pertain to the structure of a general ethical or legal code. In the case of contracts we are concerned with obligation and permission at a more specific level – for example, x is obliged *to* y to do Φ or x is permitted *by* y to do Φ. Our view here is that these specific obligations and permissions depend on protection under the general legal system in force. We regard this protection to be in the form of the possibility of taking legal action if the terms of the contract are violated. We abbreviate party x taking legal action against party y as $LA(x, y)$. Thus, our interpretation of x's obligation to y to do Φ is that y is permitted to take legal action against x if Φ does not occur. We abbreviate this as follows:

$$O(x, y)\Phi ::= \Diamond_D \neg\Phi \rightarrow L(y, x)$$

The symbol $O(x, y)\Phi$ will be read "x has the obligation to y to see to it that Φ". Usually, x will be an agent of an action contained in Φ, though this is not required. For instance, Φ might be performed by someone else sub-contracted by x. The concept of contractual permission is slightly less direct than for contractual obligation. We will use the notation,

$$P(x, y)\Phi$$

to indicate that "x permits y to bring about Φ". We begin with the observation that this generally presupposes that y would otherwise be prohibited from doing (bringing about) Φ, which is to say that x would be permitted legal action against y, if y did Φ. Thus, by granting permission to y to do Φ, x foregoes this right to take legal action. In symbolic form, this is summarized as follows:

$$P(x, y)\Phi ::= \neg[\Diamond_D \Phi \rightarrow LA(x, y)]$$

Reading: that x permits y to do Φ is defined as that it is not permitted for x to take legal action against y if Φ.

In the preceding section we saw that the various forms of modal operators, including the deontic modals, were logical duals of one another. This is also the case with contractual obligation and permission as we have defined it — however, with one interesting difference: the order of the agent and recipient places is reversed in the dual form. Thus,

$$P(x, y)\Phi ::= \neg O(y, x)\neg\Phi$$

$$::= \neg[\Diamond_D \neg(\neg\Phi) \rightarrow LA(x, y)]$$

$$::= \neg[\Diamond_D \Phi \rightarrow LA(x, y)]$$

Reading: x permits y to Φ is defined that y is not obligated to x not to Φ. The subsequent substitutions lead to the definition of contractual permission given previously. One additional aspect needs to be considered. In the contracts discussed so far, the enforcement of the contract was an implicit recourse to legal action. However, in certain contracts, this enforcement is made explicit in the form of a penalty clause indicating some other action to be taken. We will indicate such penalty clauses by adding an additional place in the contractual obligation and permission operators, separated by a "/". [5] Thus,

$$O(x, y)\Phi/\Psi ::= \Diamond_D[\neg\Phi \rightarrow \Psi].$$

The previous syntax is thus a special case of this, where Ψ = LA(y,x):

$$O(x, y)\Phi ::= O(x, y)\Phi/LA(y, x)$$
$$::= \Diamond_D \neg\Phi \rightarrow LA(y, x)$$

While explicit penalty clauses are fairly common in the case of contractual obligation, they are rare for contractual permission. Nevertheless, for the sake of completeness and symmetry, we offer the following definition:

$$P(x, y)\Phi/\Psi ::= \neg\Diamond_D[\Phi \rightarrow \Psi]$$

Letting Ψ be LA(x,y), the earlier definition follows as a special case:

$$P(x, y)\Phi ::= P(x, y)\Phi/LA(x, y)$$
$$::= \neg[\Diamond_D[\Phi \rightarrow LA(x, y)]$$

15 Summary of the Language CANDID

15.1 Syntax of CANDID

Corresponding to each type, a, the intension of that type will be a new type, written <*s, a*> The set of types (i.e. Type) is defined recursively as follows:

v is a type
e is a type
c is a type
n is a type

[5] Note: Von Wright also uses a slash notation resembling this, but with a different interpretation

t is a type
If a and b are types, then $<a,b>$ is a type.
If a is any type, then $<s,a>$ is a type.

Basic Expressions: The basic expressions of CANDID consist of:

Constants: for each type a, the set of *constants of type a* denoted Con_a. (Names for particular constants follow the conventions described earlier.)

Variables: for each type, a, the set of *variables of type a*, denoted Var_a. (Names for variables follow the conventions described earlier.)

Terms: for each type a, the set of *terms of type a* denoted $Term_a$, defined recursively as follows:

— if $\alpha \in Con_a$ then $\alpha \in Term_a$
— if $\alpha \in Var_a$ then $\alpha \in Term_a$
— if $\beta_1, \ldots, \beta_n$ are terms of types $x_1, \ldots, x_n$
respectively and Φ is an operation of type $<x_1, \ldots, <x_n, a>>$ then
$\Phi(\beta_1, \ldots, \beta_n) \in Term_a$.
— if $\mu \in Var_a$ and $\Phi \in ME_{<a,t>}$, whose only unbound variable is μ, then $\iota\mu\Phi \in Term_a$.

Formation Rules of CANDID

Syn_{CANDID} 1: Every term of type a is in ME_a
Syn_{CANDID} 2: If $\alpha \in ME <a, b>$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.
Syn_{CANDID} 3: If $\alpha \in ME_a$ and μ is a variable of type b, then $\lambda\mu\alpha \in ME_{<b,a>}$
Syn_{CANDID} 4: If α and β are both in ME_a, then $[\alpha = \beta] \in ME_v$.
Syn_{CANDID} 5: If $\Phi \in ME_v$ and $\mu \in Var_a$ then $\iota\mu\Phi \in ME_a$

Syn_{CANDID}6-10: If Φ and Ψ are in ME_v, then so are:
Syn_{CANDID} 6: $\neg\Phi$
Syn_{CANDID} 7: Φ & Ψ
Syn_{CANDID} 8: $\Phi \vee \Psi$
Syn_{CANDID} 9: $\Phi \rightarrow \Psi$
Syn_{CANDID} 10: $\Phi \leftrightarrow \Psi$

Syn_{CANDID} 11-12: If $\Phi \in ME_v$ and μ is a variable of any type, then
Syn_{CANDID} 11: $\forall\mu\Phi \in ME_v$
Syn_{CANDID} 12: $\exists\mu\Phi \in ME_v$

Syn_{CANDID} 13: if $\mu \in Var_t$ and $\Phi \in ME_v$ then $R\mu\Phi \in ME_v$
Syn_{CANDID} 14: If Φ and Ψ are in ME_v, then $\Phi\ T\ \Psi \in ME_v$
Syn_{CANDID} 15: If $\Phi \in ME_v$, then $\bigwedge[\Phi] \in ME_{<s,v>}$
Syn_{CANDID} 16: If $\Phi \in ME_{<s,v>}$, then $\bigvee \Phi \in ME_v$
Syn_{CANDID} 17: If Φ and Ψ are in ME_v, and α is a term of type e, then $[\Phi I \alpha \Psi] \in ME_v$.

Syn_{CANDID} 18-19: If $\Phi \in ME_v$, then
Syn_{CANDID} 18: $\Diamond_D \Phi \in ME_v$
Syn_{CANDID} 19: $\Box_D \Phi \in ME_v$

15.2 Semantics of CANDID

A *model* for CANDID is an ordered tuple $M = <D,\ C,\ N,\ T,\ W,\ F>$ where D, C, N, T and W are non-empty sets, that assign an appropriate denotation to each (non-logical) constant of CANDID relative to each pair $<i,j>$, for i ∈ W and j ∈ T. Thus "F($<i,j>$, α) = γ" asserts that the denotation of the constant α in the possible world i at time j is the object γ.)) The set of *possible denotations* of type a, written D_a, is defined as follows (where a and b are any types):

$D_e = D$
$D_c = C$
$D_n = N$
$D_t = T$
$D_v = \{True, False\}$
$D_{<a,b>} = D_{b_{D_a}}$
$D_{<s,a>} = D_{a_{WxT}}$

(where WxT is the set of all world, time point pairs, i.e. the set of all indices $<i,j>$).

Semantic Rules The semantic rules of CANDID define recursively for any expression α, the extension of α with respect to model M, i ∈ W, j ∈ T and value assignment g, written $Ext_{M,i,j,g}(\alpha)$ as follows:

Sem_{CANDID} 1: If α is a non-logical constant, then $Ext_{M,i,j,g}(\alpha) = F(\alpha)(¡i,j¿)$. (That is, the extension of α at $<i,\ j>$ is simply the result of applying the intension of α, which is supplied by F, to $<i,\ j>$).

Sem_{CANDID} 2: If α is a variable, then $Ext_{M,i,j,g}(\alpha) = g(\alpha)$.

Sem_{CANDID} 3: If $\alpha \in ME <a,b>$ and μ is a variable of type b, then $Ext_{M,i,j,g}(\lambda\mu\alpha)$ is that function h with domain D_b such that for any object

x in that domain, $h(x) = Ext_{M,i,j,g'}(\alpha)$, where g' is that value assignment exactly like g with the possible difference that $g'(\mu)$ is the object x.

Sem_{CANDID} 4 If $\alpha \in ME_{<a,b>}$ and $\beta \in ME_a$, then $Ext_{M,i,j,g}(\alpha(\beta))$ is $Ext_{M,i,j,g}(\alpha)$ $(Ext_{M,i,j,g}(\beta))$ (i.e., the result of applying the function $Ext_{M,i,j,g}(\alpha)$ to the argument $Ext_{M,i,j,g}(\beta)$).

Sem_{CANDID} 5: If α and β are in ME_a, then $Ext_{M,i,j,g}(\alpha = \beta)$ is $True$ if and only if $Ext_{M,i,j,g}(\alpha)$ is the same as $Ext_{M,i,j,g}(\beta)$.

Sem_{CANDID} 6: If $\Phi \in ME_v$, then $Ext_{M,i,j,g}(\neg\Phi)$ is $True$ if and only if $Ext_{M,i,j,g}(\Phi)$ is $False$, and $Ext_{M,i,j,g}(\neg\Phi)$ is $False$ otherwise.

Sem_{CANDID} 7: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi\&\Psi]$ is $True$ if and only if both $Ext_{M,i,j,g}(\Phi)$ and $Ext_{M,i,j,g}(\Psi)$ are $True$.

Sem_{CANDID} 8: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \vee \Psi]$ is $True$ if and only if either $Ext_{M,i,j,g}(\Phi)$ is $True$ or $Ext_{M,i,j,g}(\Psi)$ is $True$.

Sem_{CANDID} 9: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \rightarrow \Psi]$ is $True$ if and only if either $Ext_{M,i,j,g}(\Phi)$ is $False$ or $Ext_{M,i,j,g}(\Psi)$ is $True$.

Sem_{CANDID} 10: If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}[\Phi \leftrightarrow \Psi] = True$ if and only if either both $Ext_{M,i,j,g}(\Phi)$ and $Ext_{M,i,j,g}(\Psi)$ are both $True$ or both $False$.

Sem_{CANDID} 11: If $\Phi \in ME_v$ and μ is a variable of type e, then $Ext_{M,i,j,g}(\forall\mu\Phi) = True$ if and only if $Ext_{M,i,j,g'}(\Phi) = True$ for all value assignments g' exactly like g except possibly for the value assigned to μ.

Sem_{CANDID} 12: If $\Phi \in ME_v$ and μ is a variable of type e, then $Ext_{M,i,j,g}(\exists mu\Phi) = True$ if and only if $Ext_{M,i,j,g}(\Phi) = True$ for some value assignment g' exactly like g except possibly for the value assigned to μ.

Sem_{CANDID} 13: For α a term of type t, and $\Phi \in ME_v$, then $Ext_{M,i,j,g}[R\alpha\Phi] = True$ if and only if for time $j' = F(\alpha)$, and $j' < j$, $Ext_{M,i,j',g}[\Phi] = True$.

Sem_{CANDID} 14:. If Φ and Ψ are in ME_v, then $Ext_{M,i,j,g}(\Phi\ T\ \Psi)$ is $True$ if and only if $Ext_{M,i,j,g}(\Phi)$ is $True$ and $Ext_{M,i,j',g}(\Psi)$ is $True$ for the unique j' such that $j < j'$ and for all j", either not $j < j'' < j'$ or $j'' = j'$.

Sem_{CANDID} 15: If $\alpha \in ME_{\alpha}$, then $Ext_{M,i,j,g}(\bigwedge \alpha)$ is that function h with domain WxT such that for all <i', j'> in WxT, $h(< i', j' >)$ is $Ext_{M,i',j',g}(\alpha)$.

Sem_{CANDID} 16: If $\alpha \in ME_{s,a}$, then $Ext_{M,i,j,g}(\bigvee \alpha)$ is $Ext_{M,i,j,g}(\alpha) = (<i,j>)$ (that is, the result of applying the function $Ext_{M,i,j,g}(\alpha)$, to the argument $<i, j>$).

Sem_{CANDID} 17: If Φ and Ψ are in ME_v, and α is a term of type e, then $Ext_{M,i,j,g}(\Phi I \alpha \Psi) = True$ if and only if $Ext_{M,i,j,g}(\Phi) = True$ and $Ext_{M,i',j,g}(\Psi) = True$ for some i' just like i except that i' lacks the interference of agent α.

Sem_{CANDID} 18: For $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Diamond_D \Psi) = True$ if and only if $Ext_{M,i',j',g}(\Psi) = True$ for some $i' \in W_D$ and some $j' \in T$.

Sem_{CANDID} 19: For $\Phi \in ME_v$ then $Ext_{M,i,j,g}(\Box_D \Psi) = True$ if and only if $Ext_{M,i',j',g}(\Psi) = True$ for all $i' \in W_D$ and all $j' \in T$.

Additional Primitive and Derived Predicates and Operations

For Domain C (character strings) None

For Domain N (numbers)

1. type: $<n, <n, v>>$
   $[\alpha < \beta]$ primitive
   $[\alpha \leq \beta] ::= [\alpha < \beta] \vee [\alpha = \beta]$
   $[\alpha > \beta] ::= \neg[\alpha \leq \beta]$
   $[\alpha \geq \beta] ::= [\alpha > \beta] \vee [\alpha = \beta]$
   $[\alpha \neq \beta] ::= \neg[\alpha = \beta]$

2. type: $<n, <n, n>>$
   $[\alpha + \beta]$ primitive
   $[\alpha - \beta]$ primitive
   $[\alpha / \beta]$ primitive
   $[\alpha * \beta]$ primitive
   $[\alpha * * \beta]$ primitive

For Domain T (times)

1. type: <*t*, <*t*, *v*>>
   $[\alpha < \beta]$ primitive
   $[\alpha \leq \beta] ::= [\alpha < \beta] \vee [\alpha = \beta]$
   $[\alpha > \beta] ::= \neg[\alpha \leq \beta]$
   $[\alpha \geq \beta] ::= [\alpha > \beta] \vee [\alpha = \beta]$
   $[\alpha \neq \beta] ::= \neg[\alpha = \beta]$
   $NEXT(\alpha, \beta) ::= (\alpha < \beta) \& \forall t[(t \neq \alpha) \& (t \neq \beta) \rightarrow \neg[\alpha < t < \beta]]$

2. type: <*t*, <*t*, <*t*, *v*>>>
   $Span ::= \lambda x \lambda y \lambda z[z \geq x \& z \leq y]$

3. type: <<*t*, *v*>, *t*>
   $Beg ::= \lambda x \iota y \exists z[x = Span(y, z)]$
   $End ::= \lambda x \iota z \exists y[x = Span(y, z)]$

4. type: <<*t*, *v*>, *v*>
   $PT ::= \lambda x \lambda y[(Beg(x) \geq Beg(y)) \& (End(x) \leq End(y))]$

5. type: <*n*, <*n*, <*n*, <*t*, *v*>>>>
   $Date(\alpha, \beta, \gamma)$ (primitive)

6. type: <*n*, <*n*, <*t*, *v*>>>
   $Mo(\alpha, \beta)$ (primitive)

7. type: <*n*, <*t*, *v*>>
   $Yr(\alpha)$ (primitive)

8. type: <<*t*, *v*>, *n*>
   $Dur(\alpha, \beta)$ (primitive)

9. type: *t*
   Day (primitive, measurement standard individual)

10. type: <<*t*, *v*>, <*v*, *v*>>
    $RT(\alpha)\Phi ::= \forall t(t \in \alpha) \rightarrow R(t)\Phi$
    $RD(\alpha)\Phi ::= \exists \mu(PT(\mu, \alpha) \& R(t)\Phi$

11. type: <*t*,<*v*, *v*>>
    $RB(\gamma)\Phi ::= \exists \mu[(\mu \leq \gamma) \& RD(Span(\gamma, \mu))\Phi]$

For Domain D (deontics)

12. type: $<e, <e, v>>$
    LA (primitive)

13. type: $<e, <e, <v, v>>>$
    $O(\mu,\nu)\Phi ::= \Diamond_D\neg\Phi \rightarrow LA(\mu,\nu)$
    $P(\mu,\nu)\Phi ::= \neg O(\mu,\nu)\neg\Phi ::= \neg[\Diamond_D\Phi \rightarrow LA(\mu,\nu)]$

14. type: $<e, <e, <v, <v, v>>>>$
    $O(\mu,\nu)\Phi/\Psi ::= \Diamond_D\neg\Phi \rightarrow \Psi$
    $P(\mu,\nu)\Phi/\Psi ::= \neg[O(\mu,\nu)\neg\Phi/\Psi ::= \neg[\Diamond_D\Phi \rightarrow \Psi$

References

[CK73] C.C. Chang and H.J. Keisler, *Model theory*, A. Heyting, et al. (eds.) Studies in Logic and The Foundations of Mathematics, vol. 73, North-Holland and American Elsevier Publishing Co. Inc., Amsterdam and New York, NY, 1973.

[Cre73] M.J. Cresswell, *Logics and languages*, Nethuen & Co. Ltd., London, UK, 1973.

[Dow78] D.R. Dowty, *A guide to Montague's PTQ*, Indiana University Linguistics Club, 1978.

[Fre93] Gottlob Frege, *Grundgesetze der arithmetik*, Verlag Hermann Pohle, Jena, Germany, 1893, Partial translation as *The Basic Laws of Arithmetic* by M. Furth, Berkeley: U. of California Press, 1964.

[KMM80] D. Kalish, R. Montague, and G. Mar, *Logic: Techniques of formal reasoning*, second edition ed., Harcourt Brace Jovanovich, Inc., New York, 1980.

[Kri63] S. Kripke, *Semantical considerations on modal logics*, Acta Philosophica Fennica, Modal and Many-Valued Logics (1963), 83–94.

[Lee80] Ronald M. Lee, *CANDID: a logical calculus for describing financial contracts*, Ph.D. thesis, The Wharton School, University of Pennsylvania, Philadelphia, PA, 1980, Available as WP-80-06-02, Department of Operations and Information Management (née Decision Sciences).

[Mon02] Richard Montague, *The proper treatment of quantification in ordinary English*, Formal Semantics: The Essential Readings (Barbara H. Partee Paul Portner, ed.), Blackwell Publishers, 2002, pp. 17–34.

[Res75] N. Rescher, *A theory of possibility*, University of Pittsburgh Press, 1975.

[RU71] N. Rescher and A. Urquhart, *Temporal logic*, Springer-Verlag, 1971.

[vF71] B.C. van Fraasen, *Formal semantics and logic*, Macmillan, 1971.

[VW65] G.H. Von Wright, *And next*, Acta Philosophica Fennica **Fasc. XVIII** (1965), 293–301.

[VW67] ———, *The logic of action – a sketch*, The Logic of Decision and Action (N. Rescher, ed.), University of Pittsburgh Press, 1967, pp. 121–136.

[VW68] ———, *An essay in deontic logic and the general theory of action*, Acta Philosophica Fennica **Fasc. XXI** (1968).

[Wit21] Ludwig Wittgenstein, *Tractatus logico-philosophicus*, Routledge and Keegan Paul, 1921, English translation by D.F. Pears & B.F. McGuinness, 1961.

[Wit58] ———, *Philosophical investigations*, third ed., Macmillan, New York, NY, 1953/1958, Translated by G.E.M. Anscombe.

CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part II: Formal Description of Economics Actors and Objects

Ronald M. Lee

Florida International University (FIU), Miami, FL, USA, r.lee@fiu.edu

Abstract. The formal language CANDID is presented as a knowledge representation formalism for artificially intelligent decision support systems. The language is specifically oriented to representation of concepts in finance, commerce and administration. Later parts of the paper demonstrate the application of CANDID to explication of corporate entities and contractual objects, as well as to various concepts in elementary finance.

1 Introduction

The purpose in this part is to illustrate the use of CANDID to the formal description of principal actors and objects of economic activity. This step contributes to the larger goal of formalizing the legal and accounting aspects of contracting and commerce that they may be subjected to a system of mechanical inference. Applications of such a system include electronically assisted contract negotiation, contract monitoring and other aspects of contract management. The concepts that appear in such applications range from the mundane and commonplace, e.g., nuts, bolts, to the complex and esoteric, e.g., partially allocated costs, sale-leaseback agreements, the U.S. Securities and Exchange Commission Regulations.

The job of a formal language for describing such concepts is to render them unambiguous down to a limited set of primitive concept that are consensually understood by all parties using the language. A computer system using this language could therefore aid in rectifying definitional misunderstandings between disagreeing parties. Likewise, as an aid to individual decision making, it can explain any of its inferences in step by step elementary terms.

A critical factor, however, is that the language be based on primitive concepts that are clearly and unambiguously understood by all its users. Subsequent definitions based on these elementary terms can then be as intricate as necessary without the danger of magnifying an elementary ambiguity. A fundamental issue here is the so-called "identification of particulars", of having

consensual recognition and labeling of the individual entities described by the language. Strawson [Str59] argues that the proper basis for such identification is the locatability of these entities in a spatial/temporal framework. Thus, for instance individual people are locatable in space/time in that they are born at a particular place and time, and have continuity in space and time until their deaths. Given sufficient factual data about a person's whereabouts throughout time, an arbitrary group of observers could presumably agree as to the identification of this individual (e.g., whether it were really an actual person, or multiple persons, an imaginary person, etc.).

Phenomena that do not have continuity in space and time are prone to much more disagreement of identification. Consider for example Beethoven's 9th Symphony. Is there one unique referent to this name, or many? We may individuate versions of this symphony by its reproductions on paper or specific performances by orchestras, but in both cases we re-case it into a representation locatable in a space/time framework. Textual works present a similar difficulty. A more modern example is a computer program, for instance SPSS (statistical package for the social sciences), as an arbitrary example. There have been numerous versions of this program and hundreds of computer installations have one of these versions . Further, at any given installation, more than one copy of the program may be executed in the machine's memory at a given time.

The problem of individuation becomes especially important when we consider not just information objects, like symphonies and computer programs, but contractual objects, like notes, bonds, stocks, options, licenses, insurance policies, etc. Clearly, it is of critical importance for a company to know it has a certain right or obligation. Indeed, it is precisely because of this problem of identification that signed documents play such an important role in contractual transactions: the signed document represents the agreement in a form locatable in space and time.

As mentioned earlier, the goal here is to formally describe the principle actors and objects of economic activity. Our criterion for formalization will be the unique identification of such entities in the spatial/temporal framework. If we consider only persons as economic actors, and physical objects as economic objects, the problem is trivial: both types of entities are locatable in space and time. However, another common type of economic entity (at least in Western societies) is a corporation. A corporation is more problematic from this perspective since it has no essential physical reality: no one of its assets, including is buildings, nor any one of its employees nor any of its executives or board members nor any one of its stockholders is essential to the identification of the corporation. Any one of these may change or be removed from the corporation, and the identity of the corporation can still continue.

The objects of economic activity, i.e., the things that are traded, present analogous problems for formal description. Money for instance is a key object of exchange. Yet money is no longer uniquely represented by physical objects

such as coins and bills, but often appears merely as magnetic records in bank accounts, these, like computer programs, lose the easy location in a unique place at a given time. Information objects, such a recorded music, printed texts and computer programs were already mentioned as presenting a problem for identification. Such objects present an interesting legal problem in that they can be "stolen" (copied) without removal of the original. (Our notion of theft is basically a physical one.) Computer, communications and photocopy technology are bringing the characteristics of this type of object to prime economic importance.

One other type of non-physical economic object was also already cited: contractual objects. Signed documents have historically provided these types of objects with an easy physical identifiability. However, in most concentrated centers of trading in contractual objects, namely commodity, bond and stock exchanges, there is a definite move towards automation of records and transactions, so that there to the identifiability of such objects becomes problematic.

1.1 Legal Framework

Concepts of economic actors and objects are defined win a general legal framework which to a certain extent varies from one country to the next. The perspective taken here is an essentially capitalistic one, where corporations, independent, though perhaps regulated by government, play a major economic role. Legal definitions and rules are all taken from United States law, the only code where the author has sufficient familiarity.

As a reference for the definitions used here, we have made use of College Business Law [RO77]. This is also suggested as a useful elementary-level reference text. However, we hope that this starting point not be taken as a boundary. The foundation concepts of contractual obligation, permission, etc. have their analogues in any society that has moved beyond a simple barter system, and it is our belief that the concepts presented here are extensible to other economic systems, whether free market, centrally controlled, or some intermediate combination.

The general legal system we mention is of course established by the ruling government, which is itself an important economic actor. However, insofar as the legal system generally reflects a long evolution in comparison to the shorter time frame of a government's transactions (i.e., the government generally cannot change the law from one transaction to the next), we prefer to separate the legal code from the government as an economic actor, and consider the government and its agencies as regulated by the law as are other economic actors. The assumption of a single legal code confines our attention here to transactions covered entirely by that code, i.e., to domestic transactions. In Part I, general permission and obligation (relative to an arbitrary set of laws and norms) was denoted as:

$\Diamond_D$ for permission (deontic possibility)
$\Box_D$ for obligation (deontic necessity)

Here to indicate our somewhat more restricted assumption to U.S. law, we designate this as:
$\Diamond_L$ for permission under U.S. law
$\Box_L$ for obligation under U.S. law

Actually, in the U.S. there are two levels of commercial laws, one at the state and one at the federal level. The scope of the federal laws pertains primarily to inter-state commerce. When we want to indicate the operation of state law, as distinguished from U.S. federal law, we will use the notation:

$\Diamond_{L,x}$ for permission under the law of state x

$\Box_{L,x}$ for obligation under the law of state x

For instance,

$\Diamond_{L,NY}\Phi$

would indicate that Φ is permitted in the state law of New York. Extension of this work to international commerce would employ still another legal level: international law. An essential difference at this level—which we avoid for present purposes—is the ultimate source of legal enforcement. In domestic transactions, the physical power of the ruling government is the ultimate enforcement of the law. At the international level, lacking a single dominating world government, such transactions are subject to the treaties and agreements established between the nations involved, and the appeal for enforcement is correspondingly complicated.

1.2 Ownership and Possession

The most fundamental concept of economics, perhaps, is that of (legal) ownership, which is designated by the predicate,

OWN(x,z)

meaning that x, and economic actor, *owns* z, an economic object. The essence of this part is to elaborate the predicates that qualify x and z. Here we adopt *OWN* as a primitive predicate. That is not to say it could not be analyzed further. For instance, there are certain differences in the concept of ownership between capitalist and communist countries, and to explicate international commerce one may want to describe these differences in terms of more elementary concepts. Another relationship between economic actors and objects

is that of possession, written

POSS(x,y)

indicating that actor x possesses object y. Again, this is a fundamental concept that we take as primitive, though its meaning might vary somewhat in other economic systems. Intuitively speaking, ownership constitutes a set of rights granted by the legal system of an actor towards an object. Possession on the other hand refers to physical custody. Usually, an actor possesses what it owns, but not always, as in the case of loans and rentals.

Actually, here in Part II, possession has only a minor role. It however figures more prominently in Part III, which discusses representation of financial contracts.

1.3 Some Further Definitions and Notation

In CANDID, predicates indicating a change in state may be defined using the connective T. Here, we will suffix the names of predicates so defined with the character "!" as a visual aid to reading the expressions. Using OWN and POSS, two such change predicates are defined as follows:

OCHANGE!(x,y,z) ::= OWN(x,z) T OWN(y, z)
PCHANGE!(x,y,z) ::= POSS(x,z) T POSS(y,z)

OCHANGE! indicates a change in ownership of the object z from x to y.
PCHANGE! indicates a change in possession of the object z from x to y.

Also, in CANDID a concept of action is defined by using so-called *TI* expressions containing the connectives T and I. Here, again, only as a visual aid, we use the suffix "!!" on the names of such predicates. Using *OWN* and *POSS*, four such action predicates may be defined:

OGIVE!!(x,y,z) ::= OWN(x,z) T [OWN(y, z) I(x) OWN(x,z)]
OTAKE!!(x,y,z)) ::= OWN(x,z) T [OWN(y, z) I(y) OWN(x,z)]
PGIVE!!(x,y,z)) ::= POSS(x,z) T [POSS(y,z) I(x) POSS(x,z)]
PTAKE!!(x,y,z)) ::= POSS(x,z) T [POSS(y,z) I(y) POSS(x,z)]

In *OGIVE!!*, x causes a change of ownership of z from x to y.
In *OTAKE!!*, y causes this same change of ownership to occur.
In *PGIVE!!*, x causes a change of possession of z from x to y.
in *PTAKE!!*, y causes this same change to occur.

2 Economic Actors

2.1 Persons, Proprietorships

The most obvious type of economic actor is individual persons, designated as:

PERSON(x).

However, in U.S. law, not all persons qualify as legitimate economic actors — minors and the insane are excluded. This more restricted set is designated *LPERSON* (legal person), defined as:

LPERSON(x) ::= PERSON(x) & AGE(x, YR) ≥ 18 & SANE(x).

Personal businesses, owned by a single individual are called *proprietorships*. In U.S. law they are not distinguished from their owner, hence,

PROPRIETORSHIP(x) ::= LPERSON(x).

2.2 Joint Ownership, Partnerships

Joint ownership is where one or more parties share equally in the ownership of an object. Essentially, the group of owners form a set which as a unit owns the object. For instance, for joint owners $x_1, \ldots, x_n$

$$OWN(z, y) \& z = \{x_1, \ldots, x_n\}$$

In U.S. law, a partnership is an economic actor consisting of such a set of equally participating persons. Hence,

$$PARTNERSHIP(z) ::= \exists x_1, \ldots, x_n LPERSON(x_1) \& \ldots \& LPERSON(x_n) \& z = \{x_1, \ldots, x_n\}$$

2.3 Private Corporations

It is at this level that the concept of an economic actor becomes philosophically challenging. A corporation is an artifice of the legal system. In the U.S., it is a "legal entity", entirely separate from and independent of it owners. Unlike proprietorships and partnerships, which are formed simply by the volition of the parties involved and have no separate legal status, a corporation is formed by a specially granted permission from the state.

Informally, this process is as follows. The group of people who want to start the corporation, called the *promoters*, submit registration information, called *incorporation papers*, and a *prospectus*, which describes the capital structure and intended function of the corporation to the governing state. If the corporation is to engage in interstate commerce, the prospectus must also be approved by the Securities and Exchange Commission (SEC).

In addition, a *certificate of incorporation* is filed by the promotors, which, if approved, is maintained by the office of the secretary of the state of incorporation. This certificate lists the corporation's principal officers, names of directors and incorporators, the total number of stock shares (each at a common value called the *par value*) and the name and number of shares held by each stockholder. The corporation cannot sell more than this initial number of shares without obtaining additional permission from the state. On acceptance by the state, this certificate becomes the corporation's *charter*.

This charter is a contractual permission by the state which, in gross terms, says the following: Stockholders have a right to vote members of the *board of directors* (at least three people) of the firm and to participate in the division of residual assets on the dissolution of the firm. The board of director's main responsibility is to appoint *officers* of the corporation, which serve as the agents of the corporation in legal transactions (e.g. engaging the corporation in contacts, hiring and management of employees). Only the officers, and the people they employ, can engage in the direct operation of the firm. Note that being a stockholder does not carry the right to participate in the management of the corporation nor to act as its agent in contracts.

To summarize, the corporation is essentially a locus of ownership, on one hand, and a locus of contractual commitment on the other. (These will define the two sides of the corporate balance sheet: its assets and its liabilities, including stockholder equity.) Changes in the things owned by the corporation and its commitments to other parties are made by the corporate officers and their employees, acting as agents. Corporate officers are appointed by the Board of Directors, which in turn are voted by the stockholders. A crucial issue from a formal standpoint, however, is the identification of this locus of ownership and commitment. If we simply dismiss it as an 'abstract object' having no spatial/temporal location, we are left with the theoretical as well as very pragmatic problem of determining when the corporation exists and the boundaries of its rights and obligations.

However, as noted above, the critical event in the formation of a corporation is the granting, by the secretary of the state of jurisdiction, of the corporate charter. This provides the creation of the corporation with a unique location in space and time. Furthermore, the corporate charter provides the corporation with a unique *corporate name* (within that state). This provides any subsequent contracts and titles of ownership with a reference to the corporate charter, and hence to a unique spatial/temporal location. Though this provides the means to identify a corporation, we have still not explained what

a corporation *is*. Clearly, it is not in itself something physical. Rather, it is a complex of contingent rights and privileges as established by the corporate laws of the state.

Let us refer to this complex as *CORP-RIGHTS*. These are granted by a particular state, and associated to a unique (within the state) corporate name. Using the notation described earlier for permission under the law of state μ, this would be

$$\diamond_{L,\mu} \textit{CORP-RIGHTS}(\nu)$$

where ν is a variable of type c, a character string indicating the name of the corporation. This describes the situation where state μ permits corporate rights associated with name ν. The type of this expression is: $<<e, <c,v>>,v>$. That is, the characteristic function of *CORP-RIGHTS* maps from character strings to truth values. The state's legal permission is a mapping from an entity (the state) and the previous expression to a truth value.

We would like to say that the corporation is simply this permission. However, if we are speaking of a certain time, t, the corporation is not simply this permission at time t, but to account for the corporation's ownership of assist, it must also include permission at previous times when the assets were acquired. Further, if the corporation is in operation it will presumably have contractual obligations to other parties. These involve evaluation of these corporate rights not only in future times, but under alternative circumstances, i.e., in other possible worlds. What we need then is to evaluate the corporate rights predicate not just currently in the 'actual' world, but across all times and in all possible worlds. This, as explained in Part I, is provided by the intension operator, "$\bigwedge$". Thus,

$$\bigwedge[\diamond_{L,\mu} \textit{CORP-RIGHTS}(\nu)]$$

The earlier expression was of type $<<e,<c,v>>,v>$. The present expression will therefore be of type $<s, <<e,<c,v>>,v>>$, i.e. adding the additional argument of type s, which is an index to a possible world / time pair. Thus, the characteristic function of this expression evaluates whether the corporate rights associated with name ν are permitted by state μ at each possible index. This, in our view, is what a corporation *is*. Hence,

$$\begin{aligned}&\textit{PRIVATE-CORPORATION}(x) ::= \\ &\quad \exists\, y \exists\, z\ \textit{STATE}(y)\ \&\ \textit{CHAR-STRING}(z)\ \& \\ &\quad x = \bigwedge [\Diamond_{L,y}\ \textit{CORP-RIGHTS}(z)]\end{aligned}$$

The discussion here has been directed towards the formal description of *private corporations*, i.e., those which are profit oriented and have stockholders who ultimately receive these profits either through dividend distribution or

dissolution of the corporation and sale of its assets. Other types of corporation might also be with a similar form of analysis. For instance, *non-profit corporations* do not have stockholders nor do they pay income tax. *Quasi-public corporations* are private corporations that provide certain public services (e.g., certain utilities, toll roads), and which are supervised by public authorities. *Public corporations*, such as cities and certain departments of local and state governments, also provide public services but are financed by the state. Each of these present certain variants on the concept of corporation we have just described.

Additionally, the concepts of state and federal governments themselves present a challenge to formal description. Indeed, they appear to be corporate-like entities, having no essential physical existence. However, in these cases one cannot appeal to a larger deontic framework as the basis for their definition, for they *are* this framework. Instead, at least in democratic societies, one would appeal to the consensus of the voting population (present and past) as a deontic basis. However, since our objectives here are primarily concerned with commercial and financial activities, we confine our discussion only to the three classes of economic actors described above: proprietorships, partnerships, and private corporations. Hence,

ECON-ACTOR(x) ::=
PROPRIETORSHIP(x) $\vee$
PARTNERSHIP(x) $\vee$
PRIVATE-CORPORATION(x).

3 Economic Objects

3.1 Physical Objects

The most obvious type of economic object are physical ones (i.e., having mass). As before, to admit these types of entities into the descriptive formalism we must be able to locate them in a spatial / temporal framework. For most types of physical objects we think of — e.g., tables, chairs, automobiles, real estate, this is unproblematic. However, when granular substances such as corn and wheat, or liquids or gases are involved, problems of identification arise because of the fluid movement of these substances. For instance, consider a contract to buy a certain volume of ocean water located at a certain latitude and longitude at a given depth, etc. Though the geographical coordinates may be certain, the particular volume (individual) of ocean water at this location is not.

The practical device that resolves this logical problem in nearly any reasonable commercial context is that of a *container*. Liquids, gases and grains are always handled in a container of some sort, and the container provides the fluid substance with a unique and stable spatial / temporal location and with that discrete identifiability. Thus, our attention here is confined to

what we call *discrete-physical-objects*, which have distinct spatial / temporal coordinates (for instance at their center of gravity) and can be uniquely identified and named. Liquids, gases, and grains are assumed always to appear within discrete containers so that the filled container is itself a discrete physical object. We are concerned here with those types of objects that can be owned. Normally, any discrete physical object can be owned; however, U.S. law specifically excludes one type, persons (slavery having been abolished). Hence we introduce a concept of LPHYS-OBJ (legal physical object), which are those that can be owned:

$$\textit{LPHYS-OBJ}(x) ::= \textit{DISCRETE-PHYS-OBJ}(x) \;\&\; \neg\, \textit{PERSON}(x).$$

3.2 Promissory Objects

If one examines the asset side of the balance sheet of a company (which lists categories of what the company owns) one of course finds a number of categories that are types of physical objects, e.g., land, plant and equipment, inventory. However, beyond these there are typically other categories that do not comprise physical objects—e.g., accounts receivable, negotiable securities, patents, licenses. These are what we call *deontic objects*. They arise as the result of a contractual permission of which the company is the beneficiary, i.e., they are 'rights' permitting the company to do something (as with licenses) or obligations of other parties to the company (as with accounts receivables, and negotiable securities). We consider the case of contractual permissions first. This is a permission by some other party, say x, to the economic actor, call it y, to do some action, say Φ. Hence,

$$P(x, y)\Phi$$

We would like to say that y *owns* this permission. However, it is not the assertion itself that y owns, but its sense or *intension* — its interpretation across all possible worlds and times*. [1] This is represented once again using the intension operator:

$$\bigwedge[P(x, y)\Phi]$$

Thus, to invent a term or the 'object' form of a permission, we call it *LPRIVILEGE* (legal privilege). Hence for economic actors x and y, and some

[1] Note: Contractual permission was defined in terms of general permission (deontic possibility), which in turn had a semantic interpretation across possible worlds and times. Thus, contractual permission is not just permission in the present but in certain future times and circumstances as qualified by Φ. Use of the intension operator here thus appears as a second lambda abstraction across indices. The purpose of this second abstraction is essentially to 'objectify' the permission, equating it with its characteristic function across possible worlds and times.

action, Φ,

$$LPRIVILEGE(z) ::= \exists\, x\, \exists\, y\, \exists \Phi\; x = \bigwedge [P(x,y)\; \Phi]$$

The treatment for the case of contractual obligations is similar. Here we will call the object form an *LPROMISE* (legal promise). Again for economic actors x and y, and action, Φ,

$$LPROMISE(z) ::= \exists\, x\, \exists\, y\, \exists \Phi\; x = \bigwedge [O(x,y)\; \Phi]$$

A *deontic object* is either one of these types:

$$DEONTIC\text{-}OBJECT(z) ::= LPRIVILEGE(z) \vee LPROMISE(z).$$

3.3 Monetary Objects

Money is obviously an important type of object in the description of commercial and financial phenomena. If we consider money only in the form of 'hard cash', i.e., coins and bills, money is simply a type of physical object:

$$CASH\text{-}MONEY(x) \rightarrow LPHYS\text{-}OBJ(x).$$

Coins and bills are obviously of a particular national currency and have a face value. Thus for instance in the U.S. predicates indicating common type of bills and coins are:

ONE-CENT-COIN(x)
FIVE-CENT-COIN(x)
TEN-CENT-COIN(x)
ONE-DOLLAR-BILL(x)
TEN-DOLLAR-BILL(x)
etc.

However, in commercial transactions, money is seldom handled at this detail level, but rather as sums of money. In this case, we add up the face values of the various coins and bills, and convert them to a common currency unit—e.g., cents and or dollars. Thus, suppose that y is a set of coins and bills, $x_1, \ldots, x_n$. Then the monetary value of y, say n, would be given by a measurement function:

$$\$(y) = n ::=$$
$$MONEY\text{-}VALUE(y,\ Dollar,\ US) = n$$

This measurement function is for tabulating face values of a sum of currency in a given nationality. Measuring one nation's currency in terms of another with this function would thus evaluate zero. So far, we have regarded money as a special type of physical object. However, the services provided by lending institutions in most countries have extended this concept of money. In the U.S. it is quite common tat a bank check is given and accepted in lieu of cash money. These checks are made against 'demand deposit' accounts in a bank, which promises to pay the payee named on the check a sum of money whose tabulated value equals the amount specified on the check. Demand deposits are thus a deontic object, indicating the obligation of the bank, say b, to the party named on the check, say x, an amount of money, assuming U.S. dollars, n:

$$DEMAND\text{-}DEPOSIT(b,x,n) ::= \bigwedge [O(b,x): \exists\, m \;\&\; \$(m) = n \;\&\; OGIVE!!(b,x,m)]$$

Reading: a demand deposit from bank b to party x in amount n for some amount of money m, whose tabulated face value in U.S. dollar is n, b gives ownership of m to x. Because checking accounts are used so often (in the U.S.), we introduce another notational abbreviation to indicate money either in the form of cash or check:

$$(\$\$(z) = n) ::= (\$(z) = n) \vee ((\exists\, b)\ (\exists\, x)\ z = Demand\text{-}Deposit(b,x,n))$$

The two abbreviations for U.S. dollars correspond to the two concepts of money used by the U.S. Federal Reserve Board to calculate the money supply. Our notation \$ corresponds to the money supply measure, M1; our \$\$ corresponds to M2.

3.4 Information Objects

Physical objects, deontic objects, and money account for most of the types of objects that are owned by economic actors and trade in commercial transactions. However, there appears to be one additional class of ownable and tradable objects not yet included: what we call *information objects*. Informally, an information object is some meaningful arrangement of symbolic patterns on a representational medium, e.g., ink on paper or electronic codes on a magnetic tape or disk. Our concept of information object corresponds to what Thompson (1981) calls "ethereal goods". He makes the excellent observation that what is distinct about this type of object is the technology of its reproduction. Thus, to him, an ethereal good is one that can be reproduced more cheaply than it can be purchased. Thus, up until the time of the photocopy machine, a book was not an ethereal good. Now there are many books that are cheaper to photo-copy than purchase from the publisher (especially

low volume technical books). Similarly, home stereo tape recorders made it cheaper to copy musical recordings than buy them. However, the innovation that really expanded the class of ethereal goods was the electronic computer. A fundamental concept in this technology is that data is easily and instantly copyable. Hence any information converted for computer storage (or indeed programs directing the processing of data) can be instantaneously reproduced (copied to another magnetic medium or sent over communication lines) at practically no cost. Since considerable labor is often expended in the original creation of such information objects, the legal problem this presents is how to protect the developer from having his/her work "stolen" — i.e., reproduced without compensation.

Our concern here, however, is only with the description of these types of objects. As we have seen, their essential characteristics are not the physical medium on which they are represented, but their reproducibility. In owning such an information object, therefore, one of course owns the physical representation medium, but more importantly, one owns rights controlling the reproduction of the object. (Thus, the copyright laws for textual material prescribe the "copy rights" of the author and publisher.) Thus, in the perspective here, the essential features of an information object are very similar to that of a license, i.e. a contractual permission from one party to another. In the case of information objects, the permitted action is a certain limited range of reproduction. Let us refer to instances of these actions as *LTD-REPRODUCTION*!!. In acquiring an information object, one therefore acquires a physical representation of the information object, plus certain rights of limited reproduction. Let k be this physical representation, x be the party acquiring the information object, and y the author or holder of the copyright of the object. Then the rights transferred — which for us *is* the information object — is defined as follows:

INFO-OBJ(z) ::=
$(\exists x)(\exists y)(\exists k)\ z = \bigwedge[P(y,x)\neg$*LTD-REPRODUCTION*$!!(k)]$

Reading: An information object, z is defined as for some parties x and y and a physical medium k, the permission of y to x to certain actions of limited reproduction of k.

4 Summary

An economic actor is defined:

ECON-ACTOR(x) ::=
LPERSON(x) $\vee$
PROPRIETORSHIP(x) $\vee$
PARTNERSHIP(x) $\vee$

CORPORATION(x).

An economic object is defined:

ECON-OBJ(z) ::=
LPHYS-OBJ(z)∨
DEONTIC-OBJ(z) ∨
MONETARY-OBJ(z) ∨
INFO-OBJ(z).

As noted earlier, the class of monetary objects comprises certain physical objects (coins and bills) and certain deontic objects (demand deposits). Also, an information object has both physical and deontic aspects to it — the physical representation of the original and the limited rights of reproduction. Thus, the above definition of economic object is redundant to this extent.

The two place predicates OWN and POSS were taken as primitive. To indicate that each is a relation between economic actors and economic objects, we have the following controlling axioms:

$OWN(x, z) \rightarrow$ *ECON-ACTOR(x) & ECON-OBJ(z).*

$POSS(x, z) \rightarrow$ *ECON-ACTOR(x) & ECON-OBJ(z).*

References

[RO77] R. R. Rosenberg and W. G. Ott, *College business law*, Shaum's Outline Series, McGraw-Hill, New York, NY, 1977.
[Str59] P. F. Strawson, *Individuals*, Anchor Books, 1959.

CANDID Specification of Commercial and Financial Contracts: A Formal Semantics Approach to Knowledge Representation, Part III: CANDID Specification of Financial Concepts

Ronald M. Lee

Florida International University (FIU), Miami, FL, USA, r.lee@fiu.edu

Abstract. The formal language CANDID is presented as a knowledge representation formalism for artificially intelligent decision support systems. The language is specifically oriented to representation of concepts in finance, commerce and administration. Later parts of the paper demonstrate the application of CANDID to explication of corporate entities and contractual objects, as well as to various concepts in elementary finance.

1 Introduction

In Part I, the formal descriptive language CANDID was developed. In Part II, this was applied to the description of the principal entities of economic activity, what we called economic actors, and economic objects. In this part, we extend the application of CANDID to consider the processes of economic activity itself, in describing the concepts of elementary finance, i.e. common types of transactions and financial instruments. We find this domain to be not only a fairly central and important one to understanding commercial activity more broadly, but also reasonably representative of the classes of conceptual problems likely to arise in efforts to formalize other aspects of business. We thus believe that analogous analyses could be applied for instance to financial accounting, cost accounting, tax law, contract law, regulatory law, etc. Again, we want to emphasize that CANDID is proposed as a *framework* for formalizing business theory, but is not intended as a theory itself. The discussion here is thus meant to be only illustrative, attempting to capture what we see as the ordinary usage and understanding of basic financial terminology and concepts. Various contemporary theories of accounting, finance and economics might therefore disagree with aspects of the analysis here. (The only responsibility we would claim for CANDID is to explicate this disagreement.)

As a general guide to what concepts should be included here, we made use of *Mathematics of Finance* [Aye63], a beginning level college primer. This is likewise suggested as an elementary background reference.

2 Additional Definitions, Notational Conventions

In Part II the concepts of an *economic actor* and *economic object* were developed. Informally, an economic actor is a legally able person or organization (proprietorship, partnership or corporation) while an economic object is a physical object (excluding persons), a contractual object (e.g. stock, bonds, licenses), a monetary object (cash or demand deposit checks) or an information object (e.g. textual materials, computer data and programs). In addition, two two-place relations between economic actors and economic objects were assumed. *OWN* (for ownership) and *POSS* (for possession). These have the following associated axioms:

OWN(x,z) → ECON-ACTOR(x) & ECON-OBJ(z).

POSS(x,z) → ECON-ACTOR(x) & ECON-OBJ(z).

Also, the notation $$ is used to indicate U.S. currency in cash or check form. E.g.

$$(m) = 158.32

indicates that the object m is a sum of money totaling $158.32. As in the earlier parts, parentheses are used for functional application arguments for predicates and functions), while square brackets are used for syntactic disambiguation. Also as previously, predicates may indicate states, changes or actions. As a visual aid, we append "!" to predicate names for changes and "!!" to names of actions. Thus, as in Part II, we have the following definitions of changes and action relating to ownership and possession.

OCHANGE!(x,y,z) ::= OWN(x,z) T OWN(y, z)
PCHANGE!(x,y,z) ::= POSS(x,z) T POSS(y,z)

OCHANGE! indicates a change in ownership of z from x to y.
PCHANGE! indicates an analogous change of possession.

OGIVE!!(x,y,z) ::= OWN(x,z) T [OWN(y, z) I(x) OWN(x,z)]
OTAKE!!(x,y,z)) ::= OWN(x,z) T [OWN(y, z) I(y) OWN(x,z)]
PGIVE!!(x,y,z)) ::= POSS(x,z) T [POSS(y,z) I(x) POSS(x,z)]
PTAKE!!(x,y,z)) ::= POSS(x,z) T [POSS(y,z) I(y) POSS(x,z)]

OGIVE!! indicates a change of ownership from x to y initiated by x, whereas *OTAKE*!! indicates the same change of ownership, but initiated by y. *PGIVE!!* and *PTAKE!!* are similarly defined for possession. One additional definition

is added for the purposes of this part, an action, *PROMISE*!!, indicating the creation and giving of a "deontic object", i.e. the (intension of) a contractual obligation or permission:

PROMISE!!(x,y,z) ::= ¬∃(p) T [∃(p) & OGIVE!!(x,y,z)]

Also, we will here need a shorthand device for describing series of conjuncts that vary only in the definition of variable names and certain numeric parameters. We call this device *iteration* and define it as follows. The notation,

$[1 \leq i \leq n]$

is read "for i from 1 to n" and is meant to assign integer values to $i = 1, 2, \ldots,$ n. Further, the variable $\mu[i]$

is replaced with respective subscripts $i = 1, \ldots,$ n, and represents n different variables. Moreover, a formula Φ, may contain several such subscripted variables $\mu, \ldots, \nu$,

$[i \leq i \leq n]$: $\Phi(\mu[i], \ldots, \nu[i]) ::=$
$\Phi(\mu_1, \ldots, \nu_1)$ &
$\Phi(\mu_2, \ldots, \nu_2)$ &
$\ldots$
$\Phi(\mu_n, \ldots, \nu_n)$.

3 Elementary Financial Concepts

3.1 Loans

Loans are a familiar and everyday concept. We think usually of a loan as letting someone use something of ours with the understanding that they will return it to us at a later time. Implicit in this notion of lending is the expectation that the borrower return the *same* object lent. We call this a *loan in substance*. For instance, renting a car or house involve loans in substance. Another type of loan, one which is especially common in business, might be called a *loan in kind*. Here the expectation is that the object returned need not be the same object, but only of the same type. For instance, loans of money, grain or oil are typically loans in kind. These two types of loans are discriminated in CANDID as follows:

LOAN-IN-SUBSTANCE!!(x,y,z,t) ::=
PGIVE!!(x,y,z) &
(∃ p) PROMISE!!(y,x,p) &

p = ⋀ [O(x,y): RD(t): PGIVE!!(y,x,z)]

Reading: x, the lender, gives y, the borrower, the object z, and y promises x that it be obligatory for y to realize sometime during time t the giving back of the same object, z.

LOAN-IN-KIND!!(x,y,Φ,t) ::=
($\exists z_1$)$\Phi(z_1)$ &
PGIVE!!(x,y,z_1) &
($\exists$ p) PROMISE!!(y,x,p) &
p = ⋀ [O(x,y): $\exists z_2(\Phi(z_2)$ & RD(t): PGIVE!(y,x,z_2))]

The reading here is similar to before except that not the object returned is not necessarily that the same one, but only one that satisfies the same predicate, Φ. Note that this second object does not necessarily exist when the *LOAN-IN-KIND*!! is realized.

3.2 Loans of Money

Loans of money are loans in kind where Φ is a money predicate. Most commonly, however, the borrower is obligated to repay a larger amount than what was borrowed, the difference being the *interest* of the loan. A loan of money with interest is thus a loan involving two kinds:

LOAN-OF-TWO-KINDS!!(x,y,Φ, Ψ*, t) ::=*
($\exists z_1$)$\Phi(z_1)$ &
OGIVE!!(x,y,z_1*)* &
($\exists$ *p) PROMISE!!(x,y,p)* &
$p = \bigwedge[O(x,y) : (\exists z_2)\Phi(z_2) \& RD(t) : OGIVE!!(x,y,z_2)]$

Here, x gives z_1 (which satisfies Φ) to y, in exchange for y's promise to later return to x some object z_2, which satisfies Ψ. Thus the thing given and the thing returned neither are the same thing, nor do they even satisfy the same predicate. This hardly seems like a loan anymore. However, in loans of money, Φ and Ψ are both money predicates that differ only in amount. For simplicity, let us assume that the currency is US dollars. Then, a loan of money with interest can be defined more specifically as follows:

LOAN-OF-MONEY1!!(x,y,n_1*,*n_2*,t) ::=*
($\exists m_1$*)* $\$(m_1) = n_1$ &
OGIVE!!(x,y,m_1*)* &
($\exists$ *p) PROMISE!!(y,x,p)* &
$p = \bigwedge$ *[O(x,y): (*$\exists m_2$*)* $\$(m_2) = n_2$
& *(RD(t): OGIVE!!(y,x,*m_2*)]*

It is more usual to specify the second amount of money as a multiple of the first. The common method is to designate a fraction, r_1, (where $100 * r_1 =$ percentage) which is the incremental portion of the first amount to be added in repayment. In this form we have:

LOAN-OF-MONEY2!!(x,y,n_1, r_1*, t) ::=*
($\exists n_2$*)*$n_2 = n_1$ ** (1+r) &*
LOAN-OF-MONEY1!!(x,y,n_1, n_2*,t).*

It is also common, at least in the US, to specify r_1 as an *annual* rate; i.e. the actual multiplier to applied to n_1, call it r_2, is determined by multiplying r_1 by the duration of t in years. Thus the load of money predicate which takes r to be an annual rate would be as follows:

LOAN-OF-MONEY3!!(x, y, n_1, r_1*, t) ::=*
($\exists r_2$*)*$r_2 = r_1$ ** Dur(t,Yr) &*
LOAN-OF-MONEY2!!(x,y,n_1, r_2*,t).*

where "*Dur*" measures the duration of years of the time span t. The interpretation so far has been that the borrower is obliged to repay the principal and interest some time within the period t. As described in this last predicate, the borrower must pay the full amount of interest irregardless of how early in this period repayment is made. While this is in fact the condition of some loans, others limit the amount of interest to apply only to the period up to the point of repayment. This form of loan would be defined as follows:

LOAN-OF-MONEY4!!(x, y, n_1, r_1, t_1*) ::=*
($\exists m_1$*) $$(m1) =* n_1 *& OGIVE!!(x,y,*m_1*) &*
($\exists$ *p) PROMISE!!(y,x,p) &*
$p = \bigwedge$ *[O(x,y): (*$\exists m_2$*)(*ιt_2*)(*$\exists t_3$*) : (End(*t_2*) < End(*t_1*)) &*
RT(t_2*) [OGIVE!!(y,x,*m_2*)] &*
Beg(t_3*)=Beg(*t_1*) &*
End(t_3*) = End(*t_2*) &*
$$(*m_2) = m * (1 +* r_1 ** Dur(*t_3*,Yr))).*

Reading: For some money, m_1, in the amount n_1, x gives this money to y; y promises that for some other money, m_2, a unique time span t_2, and some other time span, t_3, where t_2 ends before t_1 ends, and *throughout* t_2, y gives x the money, m_2, and for the time span t_3 that began with $t_1$and ended with t_2, m_2 is an amount of money equal to n_1 plus the interest on n_1 over time t_3. Note that the promise in this case involved the introduction of two time periods, t_2 and t_3, where t_2 was the (relatively short) time in which repayment is realized *throughout*, while t_3 was the time from the start of the loan to this repayment.

3.3 Simple versus Compound Interest

The interest computation in the last case is called simple interest. Often a more complex computation is used called *compound interest*. The basic effect of this is that for some time interval, called the compounding period, the interest for the period is computed and added *to the principal* for the subsequent computation. Suppose the loan is for $1000 at an annual rate of .05 for three years. Assuming a compounding period of a year, a comparison of the two methods is as follows:

End of year	Simple Interest		Compound Interest	
	Principle	Interest	Principle	Interest
1	1000	50	1000	50
2	1000	50	1050	52.50
3	1000	50	1102.50	55.12
Total†	3150.00		3157.62	

†(principle + interest)

Compounding is obviously advantageous to the lender. The computations for simple and compound interest, n, assuming principal = m, annual rate = r, total loan duration t_1, and compounding period t_2, are as follows:

$$n_{simple} = m * (1 + Dur(t_1, Yr)) \qquad n_{compound} = m * (1 + r) * * Dur(t_1, t_2)$$

While adding arithmetic complexity, compounding does not seriously complicate the descriptive complexity of the CANDID calculus. To modify the previous example to reflect compounding, one would simply change the formula for the amount of m_2 in the last line.

3.4 Present Value of a Debt

A loan or debt has value to the lender. Insofar as the promised future repayments are reasonably assured, the lender typically regards this as a component of his/her *present* wealth, even though it is only the promissory object that is actually owned. (Wealth here is taken to be the collection of things owned, according to the CANDID definition of OWN.) In business, it is very important to measure these and other forms of wealth. Since it is by the proxy of such measurements that economic objects are made numerically comparable, decision making is simplified by reducing it to arithmetic calculations and comparisons. Usually wealth is measured in monetary terms. For cash, wealth obviously is the total face amount of the currency. For physical and informational objects, wealth is typically measured as the original amount of cash paid for the object (sometimes with an adjustment for deterioration and/or obsolescence). With respect to promissory objects for *future* cash,

one might initially value them as the amount of the cash expected. However, most business and economic theorists would regard this as incorrect for two reasons:

a.) there is always some chance that the borrower may renege on the promise and the future cash may not be collectable;

b.) if the total amount to be paid were immediately available, one could invest it elsewhere (e.g. in a bank, securities, other loans) and make additional interest.

Thus a promise for future cash is usually regarded as having *less* monetary value than an equal amount in the present. The more conservative valuation is termed the *present value* of the promise. While our concern in CANDID is with the formal *description* of phenomena only, and not with *valuation* (which we see as a problem for accounting and economics), there is a commonly accepted and used method for computing the present value of future cash receipts that we feel should be mentioned here. This method involves the assumption of a rate, d, called the *discount rate*, which might be considered as a sort of counter-factual interest rate. It is the hypothetical average rate of return at which cash presently available could be invested. Considering some future cash amount, n_1, expected after a period t_1, the present value is the amount, n_2, which if invested now at the discount rate would yield money in the amount n_1. That is,

$$n_1 = n_2 * (1 + d * (Dur(t_1, Yr)))$$

hence,

$$n_2 = n_1 / (1 + d * (Dur(t_1, Yr)))$$

3.5 Partial Payments

Loans are often re-paid in a series of partial payments rather than as a lump sum. Sometimes these are of equal size and in regular intervals, though not necessarily. With respect to partial payments, it is important to distinguish the requirements of the loan from the options available to the borrower. For instance, a loan may specify payment of 36 monthly installments of a certain amount. Sometimes, however, the terms of the lay may disallow early payment. This, as we will understand it here, is not to be taken literally. Early payment is always advantageous to the lender. By such a stipulation, it is generally intended that the borrower will receive no reduction in interest due by such pre-payment. This is basically the distinction made in the predicates LOAN-OF-MONEY3!! and LOAN-OF-MONEY4!! above. As observed there,

the difference in the loan specification is that in the latter case, the amount of interest depends on the time of pre-payment. To describe loans involving partial repayments with no adjustments of interest for early payment, we can ignore the interest computation and regard the borrower's promise as a series of payments of certain pre-specified amounts $n_1, n_2, \ldots, n_k$ required on or before certain dates, $t_1, t_2, \ldots, t_k$. The borrower's obligation in this case simply covers a series of realization formulas in conjunction. For instance, suppose that on 1 January 1980 John Doe (j) borrows $1000 from his local bank (b), with repayment specified in three amounts as follows:

$250 on 31 December 1982
$500 on 31 December 1983
$300 on 31 December 1984

The CANDID description of this loan event and John's obligation are as follows:

RD(Date(1,1,1980)):
(∃m_0) $$(m_0)=1000 & OGIVE!!(b,j,m_0) &
(∃ p) PROMISE!!(j,b,p) &
p = ⋀ [O(x,y):
(∃m_1)(∃m_2)(∃m_3)
$$(m_1)=250 &
$$(m_2)=500 &
$$(m_3)=300 &
(∃t_1)(∃t_2)(∃t_3)
End(t_1)=End(Date(31,12,1982)) &
End(t_2)=End(Date(31,12,1983)) &
End(t_3)=End(Date(31,12,1984)) &
RD(t_1) [OGIVE!!(j,b,m_1)] &
RD(t_2) [OGIVE!!(j,b,m_2)] &
RD(t_3) [OGIVE!!(j,b,m_3)].

A more common formulation of a loan involves a series of equal size payments over regular intervals. The intervals most commonly used are that of a month or year which, as was noted earlier, are of varying length but nonetheless unambiguous. A loan of amount n_1 to be repaid as a series of k installments each of size n_2 in intervals of length t_1 beginning at time t_0 is described as follows:

LOAN-OF-MONEY5!!(x,y,n_1, n_2, k, t_1, t_0) ::=
(∃m_1): $$(m_1) = n_1 & OGIVE!!(x,y,m_1) & (∃ p)
PROMISE!!(x,y,p) &
p = ⋀ [O(x,y):

[i=1,k]: ∃(t[i]) ∃(m[i]):
Beg(t[i])=Beg(t_0) &
Dur(t[i]) =$i * t_1$ &
$$(m[i]=n_2 &
RD(t[i]): OGIVE!!(y,x,m[i]).

Note: as we have defined it, each repayment interval begins at the beginning of t_0, but is terminates incrmentally later by t_1 each iteration.

These descriptions provide for no reduction in interest for early payment. When that is the case, a modification analogous to that in LOAN-OF-MONEY4!! is required.

4 Financial Instruments

In the last section we looked mainly at the process of loaning money. That is, the lender gave some sum of money in exchange for the borrower's promise to pay it back in various ways. We now broaden our scope to include other financial mechanisms. As shall be seen, the notion of promise, hence promissory objects, will continue to play a central role. In approximate accordance with general usage, we refer to the promises themselves as *financial instruments*. Also in deference to general usage, the terminology of "lender" and "borrower" needs to be generalized. Broadly, we will call the "promisee" and "promissor", respectively. In more narrow contexts, these parties will be assigned more specific role names.

4.1 Leases

Leases are agreements involving monetary payments i exchange for rental or temporary possession of a physical economic object, e.g. a n apartment, house, car, truck, machine, building, land. Accountants are quick to focus on the temporariness of this possession, and when it approximates the useful life of the object, they argue that the lease effectively amounts to a sale of the object plus a corresponding financing arrangement (loan). The technicality of casting such would-be sales as leases often has certain tax advantages. Leases where the duration of possession is short relative to the object's life are termed *operating leases*. Those where the possession approximates the useful life of the object are *financial leases*. Let $p_1$be a promise (promissory object) to pay certain amounts of cash over a specified period. Then a rental for an object, z, over a period t_1, is described as follows:

LEASE1!!(x,y,z,t_1, p_1) ::=
PGIVE!!(x,y,z) &
PROMISE!!(y,x,p_1) &

(∃p_2) PROMISE!!(y,x,p_2) &
p2 = ⋀ [O(x,y): RD(t_1): PGIVE(y,x,z)].

Reading: x gives possession of z to y and y makes the promise p_1 (left unspecified, but presumably to pay money), and in addition y agrees to the promise p_2 which is the obligation to realize during t_1 the giving back of possession of z to x. Here the roles indicated as x and y are usually termed "lessor" and "lessee". Note: as described here, the lease involves two promises: p_1, to pay money, and p_2, to return the rented object. Had we wished to specify p_1, these could have been combined as a single promise. Financial leases often provide an option for the lessee to purchase the object at the end of the lease period for a usually insignificant amount, called it n_1. Such a provision is incorporated as follows:

LEASE2!!(x, y, z, t_1, p_1, n_1) ::=
PGIVE!!(x,y,z) &
PROMISE!!(y,x,p_1) &
(∃p_2) PROMISE!!(y,x,p_2) &
p2 = ⋀ [O(x,y): RD(t_1): PGIVE(y,x,z)
∨
(RD(t_1) : (∃m_1)$$(m_1) = n_1 &
OGIVE!!(y, x, m_1) TOCHANGE!(x, y, z))].

Reading: x gives to y possession of z; y promises p_1 (unspecified cash payments) to x; y also promises p_2 to x; the effect of p_2 is the obligation that: for some money m_1, in the amount n_1, either y gives to x the object z, or y gives to x the money m_1, in which case there is an (automatic) ownership change from x to y of the object z.

4.2 Options

Options as a general concept are a sort of conditional promise subject to the promisee's control. The two parties involved are sometimes distinguished as the *issuer* of the option (the promissor) and the option *holder* (the promisee). Let $Q1$ and $Q2$ be temporally unbound states of affairs, and t_1 be the span of time in which the option holds. Then the CANDID description of this is as follows. The general form of an option is the issuer's promise that if the holder acts to bring about the state of affairs $Q1$, then the issuer is obligated to act to bring about state $Q2$:

OPTION!!(x,y,Q1,Q2,t_1) ::=
(∃ p) PROMISE!!(x,y,p) &
p =
⋀ [O(x,y):

(∀t_2) [PT(t_2, t_1) &
RT(t_2): (T (Q1 (Ix) *))] →*
[(∃t_3): Beg(t_3) = End(t_2) &
RT(t_3): (T (Q2 (Iy) *))]]*

Reading: x makes some promise to y that for any time t_2 in t_1, if x brings about $Q1$ (from any state instead of any state) then it is obligatory that for some t_3 that y brings about $Q2$ (from any state instead of any state). Commonly occurring types of options are made for the purchase or sale of publicly traded stock, usually in units of 100 shares. A "call" is an option to buy 100 shares of stock at a pre-determined price. Obviously if the market price of the stock goes above this pre-set price, one can exercise the option and sell the stock in the option market at a profit. Thus, for stock in company z, at a call price of m, a call can be defined in terms of the preceding definition for an option as follows:

CALL!!(x,y,z,n_1, t_1) ::=
OPTION!!(x,y,Q1,Q2,t_1)

where

$$Q1 \leftrightarrow$$
$$[(\exists m_1)\$\$(m_1) = n_1 \ \& \ OGIVE!!(x, y, m_1)]$$

and

$$Q2 \leftrightarrow$$
$$[(\exists w)w = \{u|Stock(u, z)\} \ \& \ Count(w, Stock) = 100\& \ OGIVE!!(x, y, w)]$$

Here, $Q1$, the condition of the option, is that x gives y money in the amount n_1. $Q2$, the obligation initiated by $Q1$, is that y gives a collection consisting of 100 shares of stock in company z to$x$. A "put" is the converse of a call. It is an option to sell 100 shares of stock at a pre-established price. The holder's strategy in a put is usually that if the market price declines to below the pre-set price, the holder can buy the lower cost stock in the market and then exercise the option in order to sell it at the higher put price. The CANDID definition of a put is quite similar to a call; simply the definitions of $Q1$ and $Q2$ are interchanged:

PUT!!(x,y,z,n_1, t_1) ::=
OPTION!!(x,y,Q2,Q1,t_1)

where $Q1$, $Q2$ are defined as before. Other types of options derive from puts and calls. A "spread" is a combination of a put and a call written on the same stock and running for the same length of time. The put price is below the current market, while the call price is above it. A "straddle" is a spread where the put and call prices are equal. These would be described as conjuncts of a call and a put. A spread has two prices whereas a straddle has only one:

SPREAD!!(x,y,z,n_1, n_2, t_1) ::=
CALL!!(x,y,z,n_1, t_1) &
PUT!!(x,y,z,n_2, t_1).

STRADDLE!!(x,y,z,n_1, t_1) ::=
SPREAD!!(x,y,z,n_1, n_1, t_1).

4.3 Insurance

Insurance is a promise contingent upon some change of state in nature, rather than an action controlled by one of the parties to the promise. Let $Q1$ be a temporally unbound formula describing the event (e.g. *Earthquake()*, *Fire()*, *Flood()*), and let t_1 be the time in which the insurance is valued. Let $Q2$ be a formula describing the payment by the insurer if the event occurs. Then the general structure of an insurance policy is as follows:

INSURANCE!!(x,y,Q1,Q2,t_1) ::=
(∃ p) PROMISE!!(x,y,p) &
p =
[(∀t_2): [PT(t_2, t_1) & RT(t_2) Q1] →
⋀ [O(x,y): (∃t_3) Beg(t_3)=End(t_2) &
RD(t_3) Q2]]

Reading: x makes some promise to y that for any time t_2 on t_1 wherein $Q1$ is realized throughout, then it is obligatory following t_2 that $Q2$ be realized.

For instance, suppose party x writes insurance for party y against a fire in some building z for the appraised amount of the damage up to a maximum limit of $100,000. We assume a numeric function, *Min(nx,ny)*, which returns the smaller of its two numeric arguments, and another numeric function, *Appraisal(z)*, which returns the dollar amount of the fire damage. Then this fire insurance policy is specified as follows:

FIRE-INSURANCE!!(x,y,z,n_1, t_1) ::=
Q1 ↔ Fire!!(z) &

Q2 $\leftrightarrow$
[∃n_2) n_2 = Min(Appraisal(z), n_1)) &
(∃m_1) $$(m_1)= n_2 & OGIVE(x,y,m_1)].

4.4 Easements and Licenses

Easements and licenses are promissory objects involving permission rather than obligation. Easements are the "rights" of persons other than the owner in the use of real property (land). Presumably these rights are restricted to some particular actions or activities. If not, we would characterize the unrestricted right as possession and view the easement as a rental contract or lease. Typical kinds of easements are permissions to drive on the property, to have a building located on it, etc. These would not constitute full possession in that such other activities as extracting oil or minerals, growing crops etc. are usually not included in this permission. Let Q be the allowed activity. Then the granting of an easement by x to y on the property z over the time period t_1 is as follows:

EASEMENT!!(x,y,z,Q,t_1) ::=
(∃ p) PROMISE!!(x,y,p) &
p = ⋀ [P(x,y): RD(t_1) Q].

Reading: x makes a promise to y that y may (but doesn't have to) realize (one or more times) during the activity Q during the time period t_1.

A license, at least as we understand it here, is the general case of an easement. (Or, rather, an easement is a kind of license.) That is, it is the licenser's (promissor's) permission to the licensee (promissee) to perform certain actions that normally would be forbidden. This permission is not restricted to rights to use real property. For instance, a common type of license is for patent rights. In this case, the licenser allows the normal patent protection to be suspended for the licensee. Again, let Q be the activity permitted, and t_1 be the period of this permission. The general form of a license is then:

LICENSE!!(x,y,Q,t_1) ::=
(∃ p) PROMISE!!(x,y,p) &
p = ⋀ [P(x,y): RD(t_1) Q].

Reading: x makes a promise to y to the effect that y may do Q repeatedly during time t_1.

4.5 Debt Instruments

Loans as we discussed them in the earlier section were regarded as a particular promise (to pay cash) from one individual to another. Loans of this type,

especially when the period of the promise is less than 5 years, are usually called *notes*. *Bonds* are another type of loan. Usually these are for a period longer than five years. The promissor in these cases is generally an economic organization. e.g. a corporation or governmental body, rather than a person. The promissee (bond holder) in these cases may however, be either type of economic actor. Also, bonds usually occur as a collection of promises to a number of parties. The collection is referred to as a *bond issue*. The elements of each collective bond issue have a common agent, starting date and terms of payment. They differ in the technicality that different money is promised in each bond, though the *amount* of the money is the same, and that the recipients may be different in each case.

Two major classes of bonds are distinguished based on how the recipients are identified. A *registered bond* is one where the bond issuer maintains a record of each recipient. The bond can only be transferred by the endorsement of the issuer. A *coupon bond*, on the other hand, is payable to the "bearer". This is the more frequent form, comprising 90 % of all bonds. But the concept of "bearer" raises the interesting and potentially knotty question, "bearer of what?". Our treatment of financial instruments thus far has regarded them as abstract objects, what we have called "promissory objects". The physical representation (document) on which this promise is expressed has so far not been of importance. If we consider only the promissory object, we would view the promise to be made to some indefinite recipient who is the owner of that promise on some given date. Thus, the promissee would be indicated within the elaboration of the promise as its owner as of some future date:

$$p = \bigwedge [O(x,y): (\forall\ w): RD(t_1)\ OWN(w,p) \rightarrow (\exists m_1)\ OGIVE!!(x,w,m_1)].$$

Here the promise p is the obligation that for whoever owns p, x will give them m_1 (some money). This is however a logical anomaly, a so-called "self-referring" expression. Substitution of p in the argument of OWN here leads to an infinite regress.

In addition, there is a pragmatic problem with this definition. The promissory object, p, is merely an artifice; an abstraction without physical reality. Given that many people might claim to be the owner of the promise on the date t_1, how is the company to identify which is the real one? In the case of the coupon bonds (or any bearer bonds for that matter), the issuer generally does not keep a record of the promises. The whole point of a coupon bond is to be able to trade them without notifying the issuer. How, then, does the issuer know who to pay? The actual mechanism involved is a book containing physical coupons, one for each promised payment. These coupons operate effectively as post-dated checks of specified amounts, but with the recipient left unspecified. After any particular date is reached, the holder of this book

removes the appropriate coupon and cashes it at a bank (any bank). This physical book is thus an "authoritative document" in that its purpose is not only *informative*, containing information which can be copied as is the case with other information objects, but also *performative*, in that the promissory object in this case is identified with this unique physical object. Note that this performative aspect cannot be reproduced in a photocopy (except under false pretence). Designating this book by the variable x, the previous formula would now read:

$$
\begin{aligned}
&(\iota\ x)\text{: } AUTH\text{-}DOC(z,\ p)\ \&\\
&\quad p = \bigwedge [O(x,y):\\
&\quad (\forall\ w)\text{: } [RD(t_1)\ OWN(x,z) \rightarrow\\
&\quad (\exists m_1)\ OGIVE!!(x,w,m_1)].
\end{aligned}
$$

Thus, at least in the case of coupon bonds, any change in ownership of the promissory object must also be accompanied by a corresponding change of ownership of the coupon book. This is expressed:

$$
\begin{aligned}
&COLLATERAL\text{-}PROMISE!!(x,y,Q,t_1,z) ::=\\
&\quad (\exists\ p)\ PROMISE!!(x,y,p)\ \&\\
&\quad p = \bigwedge [O(x,y):\ RD(t_1)\ Q] \vee\\
&\quad [P\ (\forall t_2)\text{: } Beg(t_2){=}End(t_1)\ \&\\
&\quad RD(t_2)\ OTAKE!!(y,x,z)]]
\end{aligned}
$$

The promise reads as follows: for all times t_2 following t_1, it is obligated to realize during t_1 the action Q; or else it is permissible that y takes ownership of the object z from x.

4.6 Equity Instruments

Equity instruments are the various types of corporate stock. The two principal types are *common* and *preferred*. Common stock corresponds most closely with the ordinary concept of "ownership" of the corporation. Each share of common stock permits the holder to one vote in the election of the company's board of directors (usually — there have been exceptions). Beyond that, however, the stockholder has little direct influence on the firm's everyday operations nor can he/she legally dispose of any of the firm's assets without the permission of the management or board. Common (as well as preferred) stockholders are not responsible for the corporation's debts. If the firm goes bankrupt, creditors have no claim to the stockholder's personal estate. If the firm is liquidated without bankruptcy, common stockholders have a residual claim to the assets — they get whatever is left after all debts have been satisfied as well as whatever claims preferred stockholders might have.

We find this to be a quite different form of "ownership" than the others we have considered. For that reason, we have expressly *excluded* it in the

definition of our OWN predicate. While stockholders are seen to OWN their stock, they are *not* seen to OWN the corporation itself. Rather, the stock is regarded as a promise, essentially no different than the promises involved in debts, to which the corporation has a commitment.

The details of these promises are rather vague however. Roughly, they are contingent obligations on the part of the firm to eventually distribute cash dividends, and/or accumulate valuable assets within the firm which may be eventually converted to cash on liquidation. Seldom, if ever, are these commitments ever articulated however. (Certainly they exist or else the stock would have no value.) Given the vagueness and complexity of the corporation's agreement with its stockholders, we are forced (at least for the moment), to accept this a s a primitive type of promise, viz. *COMMON-STOCK*. Thus, for a corporation, c, and a stockholder, x, we would describe their relationship as follows:

$$(\exists\ p)\ PROMISE!!(c,x,p)\ \&$$
$$p = \bigwedge\ [O(x,y)\text{: } COMMON\text{-}STOCK]$$

(Recall that by the definition of *PROMISE!!*, x afterwards *OWN*s p.) Preferred stock is conceptually something of an intermediate category between bonds and common stock. It often does not have voting privileges, and sometimes is only contingently voting, e.g. only under certain adverse circumstances. In the event of liquidation, preferred stockholder's claims come after those of bond holders, but before common stock holders. Also, the nature of the firm's promise is usually more definite with preferred stock than with common, but usually contains contingency provisions not found in bonds. There is a wide range of variations written into the terms of preferred stock issues. Often there is a fixed dividend rate set, which is payable provided the firm realizes adequate earnings. Sometimes this dividend obligation is made *cumulative*, so that a missed dividend one period is added to the dividend promised for the following period. Other terms are also variously included, such as call and sinking fund provisions allowing the firm to retire this stock if it chooses. Unlike bond holders, preferred stockholders cannot legally enforce arrearages in dividends, though these dividends to take priority over dividends to common stockholders. This lack of legal enforcement is problematic in CANDID, since we have presumed that our deontic operators have the force of law. To give an example of what a preferred stock might look like in CANDID, let us assume a firm, x, writes a preferred stock to a party, y, promising a cumulative dividend interval, t_1 (for instance every year) in the amount n. Assume the stock is issued in time t_0 and that any dividends paid will be paid within t_2 (e.g. a month) time following the end of the operating interval t_1 (e.g. the fiscal year end). The notion of a dividend contingent on adequate income, would also necessitate an event predicate, *Income!(x)*,

which would test for sufficient income.

CUMULATIVE-PREFERRED-STOCK(x,y,t_0,t_1,t_2,n) ::=
(∃ p) PROMISE!!(x,y,p) &
$p = \bigwedge$ *[O*(x,y)*: [*$1 \leq i \leq$ **]:*
(∃m_1*[i]) $$(*m_1*[i])=n &*
(∃t_3*[i]) (∃*t_4*[i]):*
Beg(t_3*[i])=End(*t_0*) & Dur(*t_3*[i],*t_1*)=(i-1) &*
Beg(t_4*[i])=End(*t_0*) & Dur(*t_4*[i],*t_1*)=1 &*
(∃t_5*[i]) Beg(*t_5*[i])=End(*t_4*[i]) &*
Dur(t_5*[i],*t_2*)=1 &*
(∃t_6*[i]) Beg(*t_6*[i])=End(*t_4*[i]) &*
[RT(t_4*[i]) Income(x)]* →
[RD(t_5*[i]) OGIVE!!(x,y,*m_1*[i] &*
¬*[RD(*t_4*[i]) Income(x)]* →
[RD(t_6*[i]) OGIVE!!(x,y,*m_1*[i])].*

Reading: on each of an indefinite number of iterations, it is obligatory that for some money in the amount n, and for times t_4 (e.g. the current year), t_5 (a short period following t_4), and t_6 (an unlimited period following t_4), if there is income in t_3, x must pay the dividend during t_5; if there is no income in t_4, x must pay the dividend during t_6.

4.7 Convertibles

Certain bonds and preferred stock are "convertible". This means that the holder has the option to exchange them for the issuing company's common stock at some specified exchange rate. This option aspect of convertibles is structurally similar to that of puts and calls. We describe this convertible aspect as a *separate promise* taking the form of an option to exchange the current promise, p_1, by the company, that of the bond or preferred stock, for another promise, p_2, that of common stock. Let us assume for issuer x and holder y this option applies for the period t_1 and that y must respond within t_4 amount of time. Then, the issuance of this option would be as follows:

CONVERTIBLE-OPTION!!(x,y,p_1,p_2,t_1,t_2) ::=
PROMISE!!(x,y,p_1) &
$p_1 = \bigwedge$ *[O*(x,y)*: (∀*t_3*) (∃*t_4*): PT(*t_3,t_1*) &*
Beg(t_4*)=End(*t_3*) & Dur(*t_4*,*t_2*)=1 &*
[RT(t_3*) OGIVE!!(y,x,*p_1*)]* →
[RD(t_4*) OGIVE!!(x,y,*p_2*)]]*

Reading: x promises y that for any time t_3 during t_1, y gives back ownership of the promise p, then x is obliged to give to y the promise p_2 within the time t_4 (of length t_2) which immediately follows.

5 Concluding Remarks

This completes our list of sample financial instruments described using CANDID. The preceding was of course only a tutorial survey illustrating how CANDID can be used to represent financial and commercial concepts. As indicated in the introduction, the motivation behind the development of this calculus was to serve as a representation language for knowledge bases in artificially intelligent managerial decision support systems. Definitions such as these would therefore serve as a basis for inferencing in decision aiding applications, for instance in evaluating a firm's financial statements, evaluating financing alternatives, verification and monitoring of contracts, etc. Also, the implementation of this language in a deductive computer system would assist in the verification of the definitions. Even at this tutorial level, some of the definitions approached a level of complexity that was difficult to follow. As further, more detailed concepts are included, mental verification would become even more difficult, and the assistance of the computer in this process would be useful.

References

[Aye63] F. Ayers, Jr., *Mathematics of finance*, Shaum's Outline, McGraw-Hill, New York, NY, 1963.

Performatives, Performatives Everywhere but Not a Drop of Ink

Ronald M. Lee

Florida International University (FIU), Miami, FL, USA, r.lee@fiu.edu

Abstract. The feasibility of open, flexible electronic commerce relies heavily on the effective management of documentary procedures, i.e., the sequence by which (structured) business documents are exchanged among contracting parties. The communication of such documents is not merely the passing of information, but reflects, indeed *enacts*, the formation and discharge of commitments. Such communications are called *performative* (versus informative) in that the act of communicating itself is a social action that alters the (contractual, legal, ownership) relationship among the parties.

This paper proposes four tasks for supporting performative aspects of electronic commerce. The first is that performative documents be communicated using cryptographic protocols (including digital signatures) and the involvement of trusted third parties. A special problem is negotiable documents, which may involve the use of chip cards, specialized registries, or both. The second task is the definition of a common, publicly available language for the specification of documentary procedures, which is formal, computable and executable. We propose a formalism, called Documentary Petri Nets, for this purpose. The third task is the definition of standard business scenarios using this representation. This definition might be done on a proprietary basis, or perhaps by industry-wide user groups and/or international bodies such as the ISO (International Standards Organization) and the ICC (International Chamber of Commerce). A CASE (Computer Aided Software Engineering) tool presented in this paper, InterProcs, is designed to support undertaking this latter task by providing both a modeling platform and a testing environment for proposed documentary procedures. The fourth task is development of an architecture and a protocol for sharing these procedures among contracting parties. Three modes are suggested: globally standardized procedures; proprietary procedures; and multi-lateral coordination.

1 Performative Aspects of Commerce and Public Administration

1.1 Performatives as Social Action

Our everyday world consists of a variety of physical entities such as houses, cars and people, with corresponding physical properties, such as color, size, and location. But also in our everyday world are a wide variety of non-physical

entities, properties, and relationships. Examples include corporations, government agencies, departments in these organizations, one's position of employment, ownership (of land, of objects), contracts, promises, rights, privileges, and so on. These belong to the social aspect of our world. That aspect is putative—we believe in the existence of these things—yet, for the most part they are not directly perceptible. Though we may recognize the buildings of a corporation, the legal entity itself is invisible; it has neither color, nor shape. Likewise, we regularly acknowledge a wide variety of social properties and relationships—that Smith is a lawyer, that John and Alice are married, that Jones owns the house on the corner. Again, these are aspects of our world that we do not directly observe.

What is the basis of this invisible world? It is not material and does not arise in the way that atoms, electricity, and black holes do. Rather, it is consensual, a product of social interactions and institutions. These are conventions that make our societies work. A promise to meet for lunch or a contract to build an airport are ways to achieve coordination of human behavior. This leads us to consider the dynamics of this social world, how social entities, properties and relationships are brought about, modified, and terminated. We refer to this as *social action*.

The focus of this paper is on how social action and social coordination might be supported by computerized telecommunications networks. The goal is not only to make social coordination more efficient, reducing the effects of distance and bureaucratic red tape, but also more effective in permitting new types of inter-organizational relationships and alliances not presently feasible.

Computerized networks supporting social action are appearing in a wide variety of areas, such as automated teller machines, airline and hotel reservations systems, automated supplier ordering, and electronic financial markets. However, our objective is not simply to observe such developments, but rather to develop a deeper, more formal theory of the principles they reflect. Presently under development is a representation language and modeling platform for social action infrastructures. In the course of this project, we hope to demonstrate the design and behavior of a wide variety of computerized social agents and institutions, both private and public sector, and how they might flexibly interact and mutually adjust their behaviors. We also seek to make these systems adaptable by their human constituency.

In this paper we focus specifically on *doing* business via electronic networks: e-business as well as related e-government interactions with agencies such as customs and other regulatory authorities.

An important enabler for electronic commerce is the evolution of electronic document interchange (EDI), providing standard, computer-interpretable formats for common business documents so that many routine transaction-oriented communications can be handled directly by the parties' computers, without human intervention.

In the process of actually *doing* business, the communication of such documents is not merely the passing of information, but reflects, indeed *enacts*, the formation and discharge of commitments. Such communications are called *performative* (in distinction to informative) in that the act of communicating itself is a social action that alters the (contractual, legal, ownership) relationship among the parties. Similarly, the record of such communications constitutes evidence of the occurrence of these actions. Thus, we use the terminology *performative communication* for this model of communicating; *performative document* for the structure and content of the message; and *performative record* for its (secure) storage.

Because of the commitment consequences of performatives, careful controls are typically imposed so that performative communications are (a) not made falsely or accidentally (as is the case with marriage annulments) and (b) when made, are recognized as such (thus, the need for ceremonies, to enhance recognition of the event). More broadly, we hold that performative communications are *conventional*, and various enabling conditions must hold for them to succeed or have effect. For example, in the wedding case, the priest must be ordained, the couple must be of legal age, and so on. Most of these enabling conditions are also social, requiring additional social evidence (performative records). In this way, performatives (communications, documents, records) are linked, as a chain. In the context of business and public administration, we call these performative chains, *documentary procedures*. In certain cases, these are highly regular and routinized, such as the application for a lending card (performative document) from a public library. In other cases, the procedure may have conditional requirements depending on the situation, e.g., in a mortgage application, septic inspection. In still other cases, the procedure might be specially customized by the contracting parties, as in a letter of credit for international trading of goods.

1.2 Doing Things With Words

The linguistic concept of a performative was first introduced by Austin[1] and elaborated by Searle [Sea69] and others. Logical formalization (so-called illocutionary logic) is first presented by Searle and Vanderveken [SV85]. Early discussion of performative aspects of information technology applications include Lee [Lee80], involving data modeling for contracts and financial instruments; Flores and Ludlow[2] for office automation; Kimbrough, Lee and Ness [KLN84]; Lehtinen and Lyytinen [LL86], for information systems analysis and design, and Kimbrough and Lee [KL86], Dewitz and Lee [DL89], Dewitz [Dew92], for electronic commerce systems.

A performative is an utterance that not only conveys information but also, by its being spoken, accomplishes a socially significant act. For instance,

[1] [Aus62]
[2] [FL81]

the sentence "I now pronounce you husband and wife" when spoken by a priest during a marriage ceremony not only describes the relationship between the couple, but may actually *create* it. This example brings out several key features of performatives. One is that the state created by such an utterance is generally a social artifact. Obviously, the mere speaking of a few words has very little physical effect. Rather, when an utterance is successful it places one or more people under a socially-defined relation. Often, this involves a certain set of obligations, e.g., of fidelity, or economic responsibility.

The roles involved in a linguistic utterance are usually cast as speaker and listener. However, in the case of performatives, the listener role must be divided between *addressees* and *by-standers*. Clearly, not everybody attending the marriage ceremony becomes socially obligated by the priest's pronouncement, only the two people specifically addressed. The social contract surrounding a performative is not always overtly institutional, as is marriage. For instance, such remarks as "I promise to do the dishes tomorrow," are also performatives. Here, however, attention is limited to performatives in institutional environments. In these cases the speaker and addressee must have certain social qualifications in order for the performatives to have force. For example, only priests, ministers, ship captains, justices of the peace, etc., can pronounce marriages, and only unmarried couples of a certain age can become married. Further, apart from the broad social context that enables the performative to have force, for instance the church as an institution, there is also a narrower, 'conversational' context in which the performative must appear. For example, the marriage pronouncement must appear at a certain point near the end of the marriage ceremony, not at the beginning, nor afterwards, during the reception, and so on.

1.3 Written Performatives

Linguists generally refer to performatives as a type of *utterance*, that is, a spoken communication. What is sometimes overlooked is that written communications, too, may be performative. In these cases, however, the execution of the performative takes on a somewhat different character. In a spoken performative, the person making the performative is obviously identified as the speaker. In written performatives, the issue of authorship arises. Also, with spoken performatives the addressee hears the performative at the time it is spoken. Written communications, however, endure throughout time and so the addressee may receive the communication considerably later than when it was initially made. The question then arises of when during this interval does the performative comes into force.

These issues of authorship and timing are commonly resolved by a very simple device, namely the author's handwritten signature, accompanied by the date on which it was signed. The ritual of signing one's name to a document is so pervasive that its fundamental role is often not recognized. Indeed,

as a rough heuristic, one can usually distinguish purely informative documents from those with a performative component by whether or not it has a personal signature. For instance, printed announcements, bulletins, etc., seldom have signatures; contracts to pay money (checks, etc.) always do. The effect of the signature is roughly the declaration:

> *I hereby acknowledge that my beliefs and intentions are accurately described by this associated text.*

Signed documents, as performative instruments, also acquire a unique feature not possessed by their purely informative counterparts: the performative effect of the original signature is not carried over to its mechanical duplicates. For instance, in legal documents, such as contracts, wills, etc., when several copies are made, each must be separately signed by the author(s) to have legal validity.

The unique role of the original in written performatives has, by the way, its counterpart in spoken performatives as well: repeated playbacks of a tape recording of a spoken promise, for instance, do not create new promises. With written performatives the assumption of course is that the signature provides a unique identification of the author. However, the authenticity of the signature is seldom called into question (handwriting analysts are seldom needed in court). A more important effect is that it signals the author's declaration of personal responsibility for the associated statements. In the act of signing such a document the signer typically becomes acutely aware of its language and contents (especially if the text has been written by someone else, as in a standardized lease or loan contract), since (s)he is henceforth expected to behave in accordance with this declaration.

The social significance of this ritual, committing the signer to having the beliefs, attitudes or intentions as expressed in the document, has been accepted by nearly every literate culture for centuries. It is an extremely useful historical convention, being the hallmark of honesty and good faith in all kinds of institutional and governmental transactions and agreements. It should be noted, however, that a signature is not the only way of marking a performative document. In many cases, a special seal, stamp or sticker operates similarly, especially where the effect of the document is standardized and commonplace. Typically, these special performative symbols are designed with a special, intricate pattern that would be hard to mimic. Often, these serve effectively as the signature of an institution, rather than a single individual. Common examples are coins, bills, and postage stamps.

1.4 Legal Performatives

The law has long recognized that performative speech acts are actions, not merely statements [Dew92,DL89]. For example, an objective third party who witnesses the formation of an oral agreement can testify that a contract was formed. In fact, Tiersma [Tie86] points out that such evidence would stand up in court because the law recognizes that uttering a performative verb is a deed. Testifying what someone *said* is inadmissible, being hearsay evidence; however, testifying what someone *did* in making a performative utterance is admissible as evidence.

Many of the words in legal procedures indicate performance of legal acts, such as *offering* to sell goods, *licensing* someone to distribute a product, or *accepting* an offer. We call these *legal speech acts*. A legal speech act differs from an ordinary speech act in that it invokes the rules and conventions of the law and carries with it a certain legal force. That is, legal speech acts create obligations, permissions, and prohibitions – deontic states that are enforceable by law.

1.5 Deontic States Created by Legal Speech Acts

One of the characteristics that distinguishes performative speech acts from informative speech acts is the ability of performative speech acts to change the state of the world. Legal speech acts in commercial sales contracts obligate the contractual parties to perform the acts specified in their agreement. Thus, if a merchant offers to sell goods at a certain price, the buyer's acceptance of this offer obligates the merchant to perform a sales transaction. Obligations, permissions, and prohibitions are *deontic states*, derived from deontic logic, a form of logic concerned with normative concepts (see e.g., [All82,Cas82,Wri68] . A common function of performative documents in electronic commerce is to enact changes in deontic status[3]. The most common examples are to impose a duty or obligation; to waive an existing obligation; to permit some activity otherwise forbidden; and to impose a prohibition (see Figure 1).

1.6 Legal Processes as Formal Conversations

When those engaged in commerce cooperate to perform procedures, the actions of each agent trigger and restrict the actions of the other agents, each action creating a new state in which a limited set of subsequent actions is appropriate [SV85,WF87]). When rules and conventions govern the actions

[3] Another common type of performative is baptisms, i.e., rituals which assign a proper name to a person, company or other entity. An example is the christening of a ship, "I christen thee the Queen Elizabeth". Other categories and variations of performatives are elaborated in Searle and Vanderveken [SV85].

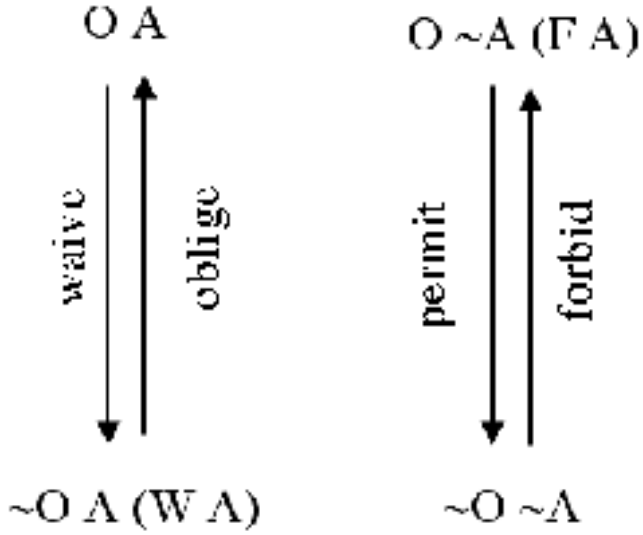


Fig. 1. Changes in Deontic Status

of agents who use words to form commitments, the procedures in this largely linguistic interaction constitute a *formal conversation*.

Our concept of a formal conversation brings together several views of linguistic interaction. From Searle and Vanderveken[4] we derive the notion of a conversation as "ordered speech act sequences that constitute arguments, discussions, buying and selling." We adopt the idea of a *consensual domain* in which agents share a common language that evolves through their activities in that domain. Few professions or trades have evolved a more elaborate language than the legal profession; legal language is, in essence, a quasi-formal language whose word meanings are fixed by common use among participants in the legal consensual domain. Formal conversations also emphasize "conversations for action" [FL81,WF87] in which agents use language to form commitments.

Formal conversations may be viewed as a kind of "language game".[5] A language game is a special context in which the use of language is governed

[4] [SV85]

[5] [Wit58]. The notion of a language game was originally proposed as a conceptual framework for organizing the contexts of usage in natural language. Our usage here is narrower, focusing on formal languages to capture the language game of legal communication. The reader should note that the language used in contract law and other legal specialties is not *ordinary* natural language but a form of specialized jargon, a kind of quasi-formal language. It is this formalized use that qualifies legal language as one of Wittgenstein's "language games."

by conventions. To understand the meaning of words in a language game, one must first understand the rules and conventions that determine how the "game" is played. In fact, the context or language game often provides the criteria for using a word; it defines "the normative aspects of certain linguistic conventions" [Fod86] that determine what an expression denotes in that game. For example, in contract law, different terms share a common, or very similar, meanings: *convey*, *transfer*, *negotiate*, *assign*, and *delegate* are legal speech acts that bring about a change in ownership. The attribute that distinguishes one of these legal speech acts from another is the object of the ownership change, in essence, the context of the use of the word. Thus, one *conveys* real estate, *transfers* tangible personal property, *negotiates* commercial paper, *assigns* contractual rights, and *delegates* contractual duties.

With this background we can now give a full definition of *formal conversation*: an ordered sequence of speech acts performed by agents who share a common language and follow prescribed rules and conventions in order to form commitments. As this definition suggests, the timing and sequencing of speech acts matter. Each speech act is an event that must occur in a prescribed order or on, by, or within a certain time. For example, in the legal procedure of forming a sales contract, an offer must precede an acceptance. Similarly, the time frame of the acceptance may be restricted: the terms of the offer may stipulate that it will expire if not accepted within 10 days.

Because timing and sequencing matter, the formal representation must employ aspects of temporal logic to represent and reason about time. Both absolute time (e.g., January 1, 1999) and relative time (e.g., event A precedes event B) must be represented so that we can infer the status of the formal conversation. That is, we must be able to determine if a contract has been successfully completed, if it is pending, or if one of the parties has failed to fulfill a legal obligation within the allotted time.

Offering and counter-offering, accepting and rejecting are all rule-governed legal speech acts within the formal conversation of contracting. When one uses these words in other conversations, their effect may not be the same; only in their "role within a certain set of social conventions or rules"[6]—within a well-defined formal conversation—do these words constitute a legal speech act capable of obligating the parties.

1.7 Legal Conversations as Documentary Procedures

One of the main reasons for the complexity of the negotiation process is the fact that parties have to know about each others' "ways of doing business" before they can start exchanging data electronically. Knowledge about the preferred way of doing business of one trading partner has to be conveyed to

[6] [KLN84]

the other; in other words, the parties have to agree upon the *documentary procedure*[7] they are going to follow.

We define a documentary procedure as *the mutually agreed upon set of steps and rules that govern the activities of all parties involved in a business transaction.* Thus, a documentary procedure controls all interactions among the roles involved. A documentary procedure stipulates which actions should be undertaken by which parties, the order in which these actions should be performed and possibly the timing constraints on the performance of these actions. Actions of parties include the sending and/or receiving of goods, documents or funds.

The need for and usefulness of documentary procedures is easy to demonstrate. Consider only a simple post-payment contract for goods. The buyer assumes that an invoice will be sent after delivery to trigger the payment obligation. The seller, on the other hand, abides by the practice that payment becomes due from the time of delivery, and does not send an invoice. Thus, the goods arrive, and the buyer does not pay, waiting for an invoice. Meanwhile the seller becomes irked, and initiates collection proceedings.

This is an example of the so-called "battle of the forms". Each party utilizes standardized documents such as a purchase order, delivery agreement, etc., which contain (typically on the backside, in small print) the terms and conditions that are their style of doing business. Unfortunately, the small print is often ignored by the receiving party.

For trade in a well-established industry area, standardized practice becomes generally accepted, and this is usually unproblematic.[8] However, in more open trading situations, that cross national, cultural or sectorial boundaries, such conflicts are much more likely to arise.

1.8 Performative Networks, Established by Umbrella Contract

Given the above framework, we can now describe a performative network more fully. A *performative network* is a telecommunications system that supports the formation of commitments between agents by providing a formal language by which they can perform legal speech acts. What makes a network performative is a set of assumptions about its use. For example, an ATM (automatic teller machine) network is performative in the sense that it

[7] It should be noted that although we call these agreements trade procedures, the principle is applicable to other societal areas than trade. The main focus of this paper however is electronic commerce which explains the term 'trade' in the definition. Other terms used to describe this concept are: trade scenarios, business scenarios and business protocols.

[8] In some cases, guidelines by international bodies such as the International Chamber of Commerce or the UNCID have been issued to diminish these ambiguities (an example is the Uniform Customs and Practices for Documentary Credits, issued by the ICC [ICC94]).

provides a formal system whereby users perform the legal speech acts of withdrawing and depositing money. In other applications, a performative network may provide a meeting place, a kind of trading room floor or market place, that imposes certain rules of discourse on its members: for example, electronic shopping (see e.g., [BL95,LW86] and electronic contracting (see e.g., [Lee80]). The sorts of performative networks that we envision for electronic commerce could also monitor the fulfillment of agents' commitments.

An informative network (e.g., the Dow-Jones News Retrieval service or any other application that provides information to subscribers) merely describes the states of participants and objects. In contrast, a performative network actually supports the performance of acts that change the status of the participants. On an informative network, unauthorized access is the primary security concern: third parties must be prevented from accessing or altering the information on the network. In contrast, on a performative network, forgery and fraud pose the greatest security risks.

Verifying the legitimacy of the acts requires verification of the agent's identity: that it was Smith who made the offer and Jones who accepted the goods. It also requires irrefutable evidence that the act itself was performed: that Smith tendered delivery of the accounting software or that Jones signaled his rejection within the trial period. Such authentication services are now commonplace in e-commerce systems.

One way to overcome this impediment is to construe the execution of a performative statement on the network as an "affirmative act" in the legal sense, similar to the act of signing a document or mailing a letter containing an offer or acceptance. Just as an acceptance conveyed in a letter is effective upon dispatch (the "mailbox rule"), a legal speech act could be seen as activated the moment the appropriate command is executed on the network. Thus, in our scenario, Jones' executing the command for entering a sale on approval is an affirmative act that signals his acceptance of the terms of the sale and that commits him to either accept and pay for the software or explicitly reject it within the 48-hour trial period.

This considerations impose legal requirements on the formation of a performative network – that it must be governed by explicit rules stipulating when and how words from the formal language can be used and what meaning the legal speech acts convey in that context. An *umbrella contract*, which all participants on the network would be required to sign, could define the rules for conducting conversations on the network, thereby defining the conventions that govern how the legal speech acts are interpreted.

2 Issues for Open Electronic Commerce

2.1 Open versus Closed Trading

While introduction of Electronic Data Interchange (EDI) promises tremendous benefits for trading partners, the costs and time of establishing clear

legal interpretations for such electronic communications can impose serious obstacles[9]. As a result, most successful EDI implementations have been realized in what could be called 'closed trading relationships', i.e., long-lasting trading relationships, involving a high number of transactions, between parties that have a high level of trust and possibly a close coordination of the parties' business processes (Table 1). In these kind of relationships, parties can gain extra benefits by closely coordinating each others' actions, thus compensating for the extra start-up costs stemming from detailed trading partner negotiations. This process is an example of business process redesign or re-engineering.

However, when the partnership is established for a limited period, covering a few transactions only and on an "at arms' length" basis, EDI linkages are seldom observed since the costs of the necessary negotiations cannot be recovered from the benefits. These shorter-term partnerships could be called 'open trading relationships' (Table 1). The main aim of our research is to contribute to the lowering of the barriers for using EDI in these open trading relationships.

Table 1. Open versus closed trading relationships.

	Open	Closed
Level of Trust	Low	High
Number of Transactions	Low	High
Duration of Relationship	Short	Long
Level of Coordination	Low	High

2.2 Automation of Documentary Procedures

The need to agree upon generally accepted documentary procedures was not discovered by the implementation of EDI; many such procedures have already been developed for paper-based commerce. This is illustrated by observing trading terms and conditions on the *back* side of documents, which supply the receiver with the knowledge needed to behave according to the terms of the business agreements. Where human involvement is high, this information might be sufficient, although experience shows that many disputes can still arise because of ambiguous formulations in such terms and conditions. In some cases, guidelines by international bodies such as the International

[9] Dewitz [Dew92] gives an example of the size of such negotiations: "At one conference on EDI law, James Pitts, a purchasing manager at R.J. Reynolds, said he spent 18 months negotiating a single trading partner agreement. That left him with only 349 other trading partners to go. "

Chamber of Commerce or the UNCID have been issued to diminish these ambiguities (an example is the Uniform Customs and Practices for Documentary Credits, issued by the ICC [ICC94]).

In electronic commerce however, when the execution of the documentary procedure is governed by automated systems, documentary procedures should be stipulated in a common formal, computable, and executable language. Such a language would allow the specification of downloadable procedures and would ease the negotiation process since all parties can express their requirements unambiguously in the same language. But this is just the first step towards electronic market relationships, since although these documentary procedures could be specified in a formal language, this still requires negotiations for each new partnership. Even if a party succeeds in creating such an agreement with one specific trading partner, if it then wants to establish EDI linkages with more partners this can lead to the existence of several slightly different documentary procedures. Clearly, this does not encourage companies to set-up new EDI linkages.

2.3 Soft Coding of Documentary Procedures

Many existing EDI applications of course embed the types of document exchange sequencing of a documentary procedure. However, these sequences are normally 'hard coded' into the application programs, as specified in the terms of the trading partner agreement – a legal, textual document. A key aspect of the architecture presented here is that documentary procedures are 'soft coded', in a declarative, rule-based form.[10] This has the virtue that they are *reusable* among different sets of contracting parties. They may be downloaded from, e.g., a central library to meet the needs of a particular contractual situation. Also significant is that such procedures can be analyzed and managed using computational tools. For example, analytical techniques can be applied to check for formal correctness (boundedness, etc.), as well as for fraud potential and other audit controls. Further, soft-coding allows for the representation of generic models that are parameterized for specific circumstances. Additionally, soft coding enables the navigation, synthesis and negotiation of procedures from different trading sectors or regulatory environments.

3 From Ink to Bits: Original, Signed Writings

Documents used in international trade are typically of a performative nature, i.e., the document itself causes a change in commitments (rights and

[10] In the terminology of programming languages, this is like the distinction between interpreted (soft) versus compiled code. In AI terms, these are declarative representations, as used for instance in expert systems.

obligations) between parties. The performative aspect of such business information poses additional security requirements on the exchange through an electronic network as compared to communications that are only informative. Even more difficult is the exchange of negotiable documents, since the electronic version needs to be treated in a different manner then their paper equivalents; the distinction between an 'original' and a 'copy' of a document disappears completely in an electronic environment.

3.1 Performative Electronic Documents

Important security requirements on the exchange of performative documents include non-repudiation, authentication, security, and integrity.

Non-repudiation: It should always be possible to prove that a specific information exchange did or did not take place. Two possible disputes can arise in this area:

- An actual information exchange took place but one of the parties denies it; or,
- Information exchange did not take place but one of the parties claims it did.

In both cases the receiver or the sender may make the false claim.

Authentication: The ability to verify the identity of the parties involved. In contract formation the most important aspect of authentication is making sure that a certain document has been sent by the party specified in the document as the sender.

Security: The ability to prevent third parties to access the information.

Integrity: The ability to rely on the underlying communication network that when a message is sent, the content is not modified during the transfer of the message.

In general, two types of solutions can be proposed to satisfy these requirements: a technical solution using cryptographic methods and an organizational solution involving trusted third parties.

Cryptographic methods employ public and private keys. It is assumed that these are distributed among parties and that the public keys of parties can be made known to others, in both case with very strict security requirements. Both sender and receiver own a private key and both keys are necessary to decrypt and/or encrypt messages. This encryption can take place in two directions: the sender can encrypt a message using his own secret key, which will make sure that people can verify that he is the original sender (only when his public key is used to decrypt this will yield a readable message). This solves the authentication problem (the 'digital signature'). Furthermore, the sender can encrypt the message using the public key of the receiver. This means that only the receiver is able to decrypt the message, since he is the only one who owns the secret key needed for this. This solves the security

problem. Since most encryption mechanisms also include the calculation of a check-sum, this technology also provides the means for the integrity checks. It should be noted however that this task should be mainly done by the electronic network provided and should be kept transparent for the sender and receiver.

The use of cryptographic methods only partly solves the non-repu-diation issue. If a message has been actually sent, the sender cannot deny this since the receiver holds a message encrypted with the secret key of the sender. Who else would be able to use this key but the sender? However, the receiver may still deny that he actually got the message but yet use the content. Conversely, if a document has not been sent, the receiver cannot provide a false message since he does not have the secret key of the alleged sender to make such a document. Of course the sender would be able to claim he did send it. Evidently, when a party would get to know the secret key of another party, this would imply that all kinds of illegal actions could take place. Therefore, very strict security constraints should be posed on the use of such keys.

The second solution to the four requirements would be to send the message through a trusted third party. For example, messages sent through a Value Added Network (VAN) may be stored by the VAN to serve as an archive to be used when a dispute arises. This would satisfy the non-repudiation and the authentication requirement, albeit with the VAN being able to access the information stored, a major drawback for an actual implementation of such a system since the VAN may be able to profit from this information. However, the security and integrity constraints cannot be solved in this manner.

The combination of these two types of solutions satisfies all these requirements. It is assumed that all messages are encrypted using the public key of the receiver (security) and the secret key of the sender (authentication or digital signature). Furthermore, the VAN keeps track of the communications through its network. Since the messages are encrypted the VAN cannot access the actual information being sent without the cooperation of the parties involved. This is impossible in a paper-based scenario, since it is hardly practicable to witness and store these communications without actually seeing the information (an envelope must be opened before the content can be copied; therefore, instruments such as registered mail can only partially fulfill the same functionality as the electronic equivalents in this respect). This would solve the non-repudiation problem. The integrity problem is solved by both the underlying network and the calculation of checksums in the encryption algorithms used.

3.2 Negotiable Electronic Documents

The implementation of negotiability is complicated by the fact that there is no distinction between originals and copies of *electronic* documents. When a document is transferred to another party, the first owner will still have an

identical copy that could also be used to exercise this right. When documents are printed on paper, several protection mechanisms are in place to guarantee the uniqueness of a document (signatures, seals, watermarks etc.). This means that if a party transfers a right by transferring an electronic document, the copy that this party retains should be made void. Three means can be chosen to solve this problem:

- There may be a technical guarantee that there is at all times only one document, i.e., the remaining copy is automatically erased beyond the control of the owner.
- A database could be maintained by an independent, trusted third party to serve as an alternative to provide information about who owns a specific right.
- The negotiable document is replaced by a non-negotiable equivalent.

A technical solution has to be used to physically erase the original when a document is sent. This can be implemented in a number of ways, such as by using smart card technology. This technology can guarantee that when a specific set of data is transferred from one smart card to the other (preferably over an electronic network), the original data are automatically deleted by the programs implemented on the smart-cards as well.

The second solution involves the introduction of a trusted third party in the form of a registry. This registry would serve as an agent for all parties involved; both the issuer of the right and the holders of the right will benefit from the registry. The registry would maintain a database containing the current holder of the right. Two functions are provided by this database. First, it provides the issuer of the right with the knowledge to verify whether a party may claim that right. Secondly, if the right is transferred, it will provide the new holder of the right with the knowledge to verify whether the seller of the right was entitled to sell it. It should be noted that the information maintained in this database should be minimized, because a party governing such a database may otherwise be able to deduce information about the trades covered and use this commercially. This minimal amount of information contains only a reference number to the actual data (and not the data itself) and the identity of the current holder.

Finally, it might be possible to circumvent the problem by using another type of document which is non-negotiable. For example, in certain kinds of documentary credit procedures the Bill of Lading may be replaced by a Waybill. Whether it is possible to make such a replacement depends of the context in which the negotiable instrument is used and the business customs and practices of those parties using the instrument.

All solutions have their merits and their disadvantages. If the negotiable document can be replaced by a non-negotiable document this is in many cases the cheapest solution. However, this is not always possible. If the negotiable instrument is a key requirement in the execution of the trade transaction, one of the other solutions should be taken. The main advantage of the smart card

solution is the freedom from dependence on third parties; when two parties decide to trade a right by exchanging the negotiable document evidencing that right, they are not depending on the performance of a third party. On the other hand, the smart-card technology poses constraints on the reachability of the parties involved and it introduces extra administrative overhead within the organizations to keep track of the smart-cards in relation with the transactions these smart-cards belong to. These disadvantages do not exist when a trusted third party is involved.

4 Computational Modeling of Documentary Procedures

4.1 Representation: Documentary Petri Nets

The representation we have developed for representing documentary procedures is based on the formalism called Petri nets [Pet62,Pet81]. The main advantage of the Petri net formalism, in addition to its capability to graphically model both concurrency and choice, is that it offers various kinds of both formal and informal analysis methods, which make Petri Nets especially suitable for modeling discrete dynamic systems.[11] In the remainder of this section, we introduce the *Documentary Petri Net* representation, an extension we developed to the classical Petri net formalism in order to satisfy the modeling requirements of documentary procedures [LB96].

A classical Petri net is a bi-partite, directed graph. It has two kinds of nodes: places (represented as circles) and transitions (represented as bars). Arcs connect places with transitions or vice versa (it is not allowed to connect two places or two transitions). The dynamic behavior of the modeled system is represented by tokens flowing through the net (represented as dots). Each place may contain several tokens (the marking of the place); a transition is enabled if all its input places (i.e., arcs exist from those places to the transition) contain at least one token. If this is the case, the transition removes one token from each input place and instantaneously produces one in each output place (i.e., an arc exists from the transition to the place). This is called the firing of a transition. The transitions in Documentary Petri Nets are labeled in order to identify the role that brings about the transition. The syntax of these labels is **Role(s) : Action**. The classical Petri nets only allow one kind of token. In order to distinguish between different types of documents, different types of places (and their tokens) have to be distinguished. In the DPN notation, these are represented as document places, drawn as a rectangle, and labeled by the document type(s).[12]

[11] [Aal92,AH02]

[12] This typing of places involves what are called colored Petri nets [Aal92]. A similar extension of the classical Petri nets are the Predicate/Transition nets, in which logical predicates are associated with transitions (Genrich and Lautenbach, 1979;

One important requirement of modeling complex scenarios is the ability to model roles as separate Documentary Petri Nets. This allows the decomposition of a documentary procedure into a number of logically separate sub-nets. This modeling style results in a clear "geographical" separation between the roles. As the role description is a sub-net in the scenario description, designers have some flexibility for experimenting with different role descriptions within the overall scenario constraints.

A state transition is enabled by receiving an information parcel, goods or funds, or the expiration of an internal timer (events). Firing a transition can lead to sending information parcel(s), goods or funds and/or setting an internal timer (actions). An example of a Documentary Petri Net model is presented in Figure 2.

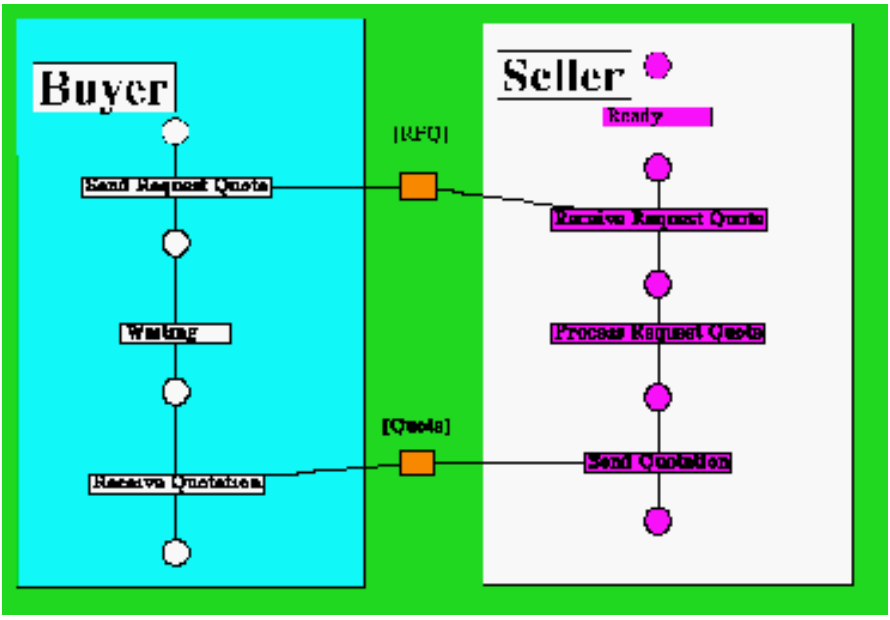


Fig. 2. Example of Documentary Petri Net

The sub-nets of the several roles may need to be combined in order to build the model of the overall documentary procedure. This can be done simply by connecting the roles at their communication points: the document and goods places. Since this process can be reversed as well, the Documentary Petri Net representation allows both a top-down and a bottom-up approach for the modeling of documentary procedures.

Genrich and Lautenbach, 1981). Documentary Petri Nets use colors and predicates to specify the different information parcel types, goods, funds and deontic states.

4.2 Prototyping Environment: InterProcs

InterProcs is a prototyping environment for documentary procedures and EDI documents developed by Lee [Lee92,Lee98]. InterProcs offers a graphical user interface with which Documentary Petri Nets can be drawn. Furthermore, since InterProcs embeds a Prolog engine, rule-bases can be added to a Documentary Petri Net model, allowing automatic reasoning about modeled documentary procedures. InterProcs cannot only be used to draw Documentary Petri Nets, it also offers the possibility to simulate and/or animate documentary procedures modeled by these nets.

The practical contribution of this research is to provide organizations with a method and a tool to define and test trade scenarios. These scenarios may be constructed either top-down or bottom-up. In the first case, an overall documentary procedure will be distributed over the individual roles. In the second case, the individual role descriptions of the parties have to be combined. In either case, the role descriptions can be distributed over multiple machines, where information parcels may be exchanged over a local or wide area network using an electronic document standard such as EDI or ebXML. This provides a realistic testing environment in which roles can be played and evaluated by different organizations.

Once tested and agreed upon, these scenarios may be stored in a public repository, governed by an international body. Since these scenarios are defined using a formal language such as the Documentary Petri Net formalism, it will be possible for organizations to then download the scenarios and execute them. During this execution the overall control on the documentary procedure is distributed among the individual organizations.

5 Protocols for Procedure Adoption

We now consider the architectural aspects of this proposal – how documentary procedures should be made available and shared among contracting parties. We distinguish three broad modalities: generally accepted standard (guideline) procedures; proprietary procedures; and multi-lateral negotiation of procedures.

5.1 Generally Accepted Standard (Guideline) Procedures

This first modality applies when there is a generally accepted style of practice to which all parties conform. This is typical of situations in which an organized market has been established, e.g., as for commodities. In this case, a 'boiler-plate' contract is available, which stipulates delivery and payment terms, handling of contingencies and, possibly, arbitration mechanisms in case of dispute. In electronic form, this entails that standardized trading procedures be made available, via a publicly accessible library, so that (role)

procedures can then be downloaded and executed by each of the parties. Although international standards for the structuring of EDI *documents* exist (i.e. UN/EDIFACT or ANSI X.12), such standardized *documentary procedures* have not been developed yet. Standardized (electronic) documentary procedures might be specified by industry groups such as EDIFICE, SWIFT or CEFIC or international trade facilitation bodies such as UNCTAD or the International Chamber of Commerce (ICC).

5.2 Proprietary Procedural Standards

The second modality is typified by situations in which one trading partner dominates the relationship, such as a large corporation, or otherwise is inflexible in its control policies, e.g., a governmental regulatory agency. The electronic interpretation of this case has (role) procedures for the other parties downloaded from a library provided by this dominant party. Such libraries expressing the 'way of doing business' of a particular party we call a *regime*.

5.3 Multi-Lateral Negotiation of Procedures

The third modality is multi-lateral, and includes situations in which several of the parties have established control policies. This is typical of situations where multiple regulatory agencies are involved. The electronic interpretation here would again be that each of these major players has their documentary procedures available in network accessible regimes. For a particular contract, each of the relevant procedures is collected and assembled. This third modality introduces the potential for conflict among control policies of the parties. In conventional trading circumstances, such conflicts (assuming they are detected!) are typically resolved in one of two ways: be negotiating compromise, or by seeking other partners.

5.4 A Messenger Model

To help cope with cases of conflicting control policies, we introduce an additional computational device, what we call a messenger. A messenger is a kind of computational agent, specialized in navigating procedure libraries or regimes, and where needed, negotiating procedural alternatives.

The notion of messenger is based on a metaphor to physical messenger services (such as UPS, Federal Express). Such physical messengers are normally charged with delivering a message or parcel. More importantly, they often make delivery of performative communications such as contractual offers (bids), legal summons, as well as payments. If obstacles arise—perhaps the recipient is not home—the messenger has some limited discretion to resolve the problem (leave parcel at neighbor's). If this is not possible, the messenger is to contact the client for further instructions. Our notion for electronic

messengers goes beyond this physical metaphor to include not only the execution of certain contractual actions, but also the navigation and (limited) negotiation of control procedures.

Here is a brief scenario. Lee, an American, lives abroad in Holland. His passport renewal is due. Normally he would need to travel to the U.S. Consulate in Amsterdam, wait in line, fill out the forms, pay the fee, etc. In all, this could easily cost an entire afternoon (if evrything goes well; if not, for instance if the passport photo is the wrong size, a return trip might be needed). Instead, Lee logs on (to InterNet) and initiates a messenger with the goal: U.S. passport renewal. The messenger contacts the U.S. Consulate's on-line regime, and identifies the procedural subset relevant to Lee's case. Since the messenger has access to the relevant personal data for Lee (including digitized passport photo), it can do most of the form-filling automatically. It then returns to Lee with a request for confirmation (plus requests for any additional data, e.g., choice of delivery mode), and release of payment for the fee. Lee gives the OK, and the messenger then returns to the Consulate and executes the transaction. The passport is sent via post or conventional messenger service. (Perhaps someday the passport itself will be electronic.)

The reader is no doubt acquainted with various other bureaucratic duties of this ilk. They include driver's license applications, voting registrations, building permits, visa applications, credit card applications, phone card applications, and so on. These illustrate the use of messengers in a uni-lateral situation. In bi- or multi-lateral cases, the messenger needs to navigate among multiple regimes and attempt to synthesize the various procedural requirements. Failing this, the messenger may attempt to locate another company or agency that offers similar goods or services, but with more compatible control requirements (either stricter or more lenient as the case may be). Or, the messenger may request a compromise in the control requirements of a current party, e.g. include the sending of an invoice where none was required. As outlined, messengers have four kinds of capabilities:

1. navigation of regimes (procedural requirements)
2. synthesis of procedures (from multiple regimes)
3. detection of procedural conflicts
4. suggestion of remedies for conflicts

These are the subject of our continuing research. To address these four areas, we are developing a generalization of the DPN representation called Procedure Constraint Grammars (PCG's). Like language grammars, which give rules of well-formedness, PCG's specify the characteristics of a family of procedures at various levels of abstraction. Just as a generative grammar for a language can produce various sentences, so too a PCG can generate particular procedures, given specified parameters. Representing regimes in this formalism, procedures specific to individual cases are extracted and presented as Documentary Petri net procedures.

Unlike language grammars, however, which are typically represented as an integrated hierarchy of rules, PCG's are organized as *constraints* on a target procedure. It is the job of the PCG constraint solver to identify a (minimal) solution procedure (according to some preference ordering of the user – e.g. minimal duration vs minimal risk). This mechanism is essential for the other three areas of messenger functionality: synthesis, conflict detection, and conflict resolution.

Presently, an initial representation of PCG's has been defined and is operational. Work continues on the procedural constraint solver design (possibly incorporating constraint logic programming, CLP). Aspects of the conflict detection problem (for static deontic rules) are addressed in Ong and Lee [OL93], using abductive reasoning. Aspects of the constraint resolution problem, again for static deontic rules, are discussed in Ryu and Lee [RL93], using defeasible reasoning.

6 Discussion and Further Research Directions

This paper has proposed four tasks for supporting performative aspects of electronic commerce.

The first is that performative documents be communicated using cryptographic protocols (including digital signatures) and the involvement of trusted third parties. A special problem is negotiable documents, which may involve the use of chip cards, specialized registries, or both.

The second task is the definition of a common, publicly available language for the specification of documentary procedures, which is formal, computable and executable. A formalism, called Documentary Petri Nets (DPN), was proposed for this purpose.

The third task is the definition of standard business scenarios using this representation. This definition might be done on a proprietary basis, or perhaps by industry-wide user groups and/or international bodies such as the ISO and the ICC. A CASE tool presented in this paper, InterProcs, intends to support these groups in this task by providing both a modeling platform and a testing environment for proposed documentary procedure designs.

The fourth task is development of an architecture and a protocol for sharing these procedures among contracting parties. Three modes were suggested: globally standardized procedures; uni-lateral coordination; and bi- or multi-lateral coordination.

We have two broad directions for future research. One of these relates to the above mentioned protocol for negotiating procedures among contracting parties with differing control requirements. This includes further refinement of the messenger model and the constraint resolution mechanism for Procedure Constraint Grammars (PCG's, [Lee02]).

The other research direction relates to normative modeling of documentary procedures: given situational goals and the parties' risk, cost and time

preferences, what constitutes a good documentary procedure? This research will result in automated verification tools that may be used to check whether a proposed documentary procedure conforms to certain requirements. Examples of automated verification tools include algorithms stemming from graph theory to detect possible dead-lock situations. Another kind of analysis has been proposed by Chen and Lee [CL03,Che92] applied to Petri Net specifications of internal accounting control structures. They have shown that the detection of undesirable patterns, for example when the ordering and payment tasks are assigned to the same person, can be performed automatically using 'audit daemons'. This approach has been extended to inter-organizational procedures in the work of Bons and Lee [Bon97,LBW01].

7 Acknowledgments

The author would like to thank Steven Kimbrough, Clive Wrigley, Yao-Hua Tan, René Wagenaar, Roger Bons, and Sandra Dewitz for many simulating discussions about performatives and deontics in electronic commerce.

References

[Aal92] W.M.P. van der Aalst, *Timed coloured Petri nets and their application to logistics*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1992.

[AH02] W.M.P. van der Aalst and K. Hee, *Workflow management: Models, methods and systems*, MIT Press, Cambridge, MA, 2002.

[All82] L.E. Allen, *Towards a normalized language to clarify the structure of legal discourse*, Deontic Logic, Computational Linguistics and Legal Information Systems (A. A. Martino, ed.), vol. II, North-Holland Publishing Company, 1982, pp. 349–407.

[Aus62] John L. Austin, *How to do things with words*, Oxford at the Clarendon Press, Oxford, England, 1962.

[BL95] James Baty and Ronald M. Lee, *Intershop: Enhancing the vendor/customer dialectic in electronic shopping*, Journal of Managment Information Systems (1995).

[Bon97] Roger W.H. Bons, *Designing trustworthy trade procedures for open electronic commerce*, Ph.D. thesis, EURIDIS at Erasmus University, Rotterdam, The Netherlands, September 1997.

[Cas82] H. N. Casteñeda, *The logical structure of legal systems: A new perspective*, Deontic Logic, Computational Linguistics and Legal Information Systems (L.E. Allen, ed.), vol. II, North-Holland Publishing Company, 1982, pp. 21–37.

[Che92] K. T. Chen, *Schematic evaluation of internal accounting control systems*, Ph.D. thesis, University of Texas at Austin, Austin, Texas, 1992.

[CL03] K.T. Chen and Ronald M. Lee, *Knowledge-based evaluation of internal accounting control systems – a pattern recognition approach*, Proceedings of American Accounting Association Conference (Honolulu, Hawaii), 2003.

[Dew92] S. D. Dewitz, *Contracting on a performative network: Using information technology as a legal intermediary*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[DL89] Sandra D.Dewitz and Ronald M. Lee, *Legal procedures as formal conversations: Contracting on a performative network*, Proceedings of the Tenth International Conference on Information Systems (Baltimore, Maryland) (Janice I. DeGross, John C. Henderson, and Benn R. Konsynski, eds.), Association for Computing Machinery, December 4-6, 1989, pp. 53–65.

[FL81] C. F. Flores and J. Ludlow, *Doing and speaking in the office*, DSS: Issues and Challenges (G. Fick and R. Sprague, eds.), Pergamon Press, London, UK, 1981.

[Fod86] J.A. Fodor, *Meaning, convention, and the blue book*, Meaning (J.V. Canfield, ed.), Garland Publishing, New York, NY, 1986, pp. 87–108.

[ICC94] ICC, *The uniform customs and practices for documentary credit procedures*, International Chamber of Commerce publication 500, January 1994, Paris, France.

[KL86] Steven O. Kimbrough and Ronald M. Lee, *On illocutionary logic as a telecommunications language*, Proceedings of the International Conference on Information Systems (San Diego, CA), December 1986, pp. 15–25.

[KLN84] Steven O. Kimbrough, Ronald M. Lee, and David N. Ness, *Performative, informative and emotive systems: The first piece of the PIE*, Proceedings of the Fifth International Conference on Information Systems (Tucson, AZ) (Leslie Maggie et al., ed.), November 28-30, 1984, pp. 141–148.

[LB96] Ronald M. Lee and Roger W.H. Bons, *Soft-coded trade procedures for open-edi*, International Journal of Electronic Commerce **1** (1996), no. 1, 27–49.

[LBW01] Ronald M. Lee, Roger W.H. Bons, and René W. Wagenaar., *Pattern-directed auditing of inter-organisational trade procedures*, Proceedings of the 1st IFIP Conference on eCommerce, eBusiness, and eGovernment (Zurich, Switzerland), 4–5 October 2001.

[Lee80] Ronald M. Lee, *CANDID: a logical calculus for describing financial contracts*, Ph.D. thesis, The Wharton School, University of Pennsylvania, Philadelphia, PA, 1980, Available as WP-80-06-02, Department of Operations and Information Management (née Decision Sciences).

[Lee92] ———, *Dynamic modeling of documentary procedures: A CASE for EDI*, Proceedings of Third International Working Conference on Dynamic Modeling of Information Systems (Noordwijkerhout, The Netherlands) (H. G. Sol, ed.), June 1992, pp. 95–123.

[Lee98] ———, *INTERPROCS: A Java-based prototyping environment for distributed electronic trade procedures*, Proceedings of the Hawaii International Conference on System Sciences, January 1998, pp. 202–209.

[Lee02] ———, *Automated generation of electronic procedures: Procedure constraint grammars*, Decision Support Systems **33** (2002), no. 3, 291–308.

[LL86] Erkki Lehtinen and Kalle Lyytinen, *Action based model of information systems*, Information Systems **11** (1986), no. 4, 299–317.

[LW86] Ronald M. Lee and George Widmeyer, *Shopping in the electronic marketplace*, Journal of Management Information Systems **2** (1986), no. 4, 21–35.

[OL93] K. L. Ong and Ronald M. Lee, *An abductive approach to detecting inconsistencies in bureaucratic systems*, monograph, Euridis, Erasmus University, Rotterdam, The Netherlands, 1993.

[Pet62] C.A. Petri, *Kommunikation mit automaten*, Ph.D. thesis, University of Bonn, Bonn, Germany, 1962.

[Pet81] J.L. Peterson, *Petri net theory and the modeling of systems*, Prentice-Hall, 1981.

[RL93] Young Ryu and Ronald M. Lee, *Formal representation of normative systems: A defeasible deontic reasoning approach*, monograph, Euridis, Erasmus University, Rotterdam, The Netherlands, 1993.

[Sea69] John R. Searle, *Speech acts*, Cambridge University Press, Cambridge, England, 1969.

[SV85] John R. Searle and Daniel Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, Cambridge, England, 1985.

[Tie86] P.M. Tiersma, *The language of offer and acceptance: Speech acts and the question of intent*, California Law Review **74** (1986), 189–232.

[WF87] Terry Winograd and Fernando Flores, *Understanding computers and cognition: A new foundation for design*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1 January 1987.

[Wit58] Ludwig Wittgenstein, *Philosophical investigations*, third ed., Macmillan, New York, NY, 1953/1958, Translated by G.E.M. Anscombe.

[Wri68] G. H. von Wright, *An essay in deontic logic and the general theory of action*, Acta Philosophica Fennica **Fasc. XXI** (1968).

EDI, XML, and the Transparency Problem in Electronic Commerce

Steven O. Kimbrough

University of Pennsylvania, Philadelphia, PA, USA,
kimbrough@wharton.upenn.edu

Abstract. Standard (that is, long-standing and currently much in use) EDI protocols (including the X12 and EDIFACT series) have repeatedly been criticized for poor design, confusing or absent semantics, and much else. Most of these criticisms are indeed on the mark. The main conclusions that the critics have drawn are also correct: business-to-business e-commerce is expensive and difficult to set up and maintain, because of shortcomings in the design concepts underlying standard EDI. Something must be done, but what?

Central to the problem is the fundamental question of semantic transparency: When A sends B a message, how does B's machine know what the message is about, what it means? Given proper standards, message meanings are determined and computers can be programmed to act appropriately to the intended message meanings. The complaint against EDI has been that proper standards cannot be made because of the misguided way in which the EDI standards are designed.

Proponents of XML have been touting XML's strengths and claiming that they overcome, or can overcome, the semantic transparency problem in e-commerce. In support of this claim, proponents point to the DTDs (or similar devices) that any XML/EDI solution would use. The claim is that semantic transparency is/can be achieved through the DTDs.

In this paper I argue that indeed the DTD mechanism offers a kind of progress on the semantic transparency problem, but that it cannot provide anything approaching a complete solution. While XML+DTDs is indeed a very promising vehicle for structuring and transporting messages for business-to-business commerce, it is not itself a semantic theory of what those messages say. We need the semantic theory. Once we have that, XML can be used to embody it for applications.

Drawing on previous work, I will present the elements of my formal semantic theory for business messaging (the "lean events theory"). With examples from this theory before us, we can get a more proper view of the semantic transparency problem (aka: the spanning problem). This is not a problem that can be made to go away entirely, but we can live with it and do commerce.

1 EDI and the Transparency Problem

The standard EDI protocols (including the X12 and EDIFACT series) are of long-standing and effective use.[1] Yet, they have repeatedly been criticized for poor design, confusing or absent semantics, and much else.[2] Many of these criticisms are indeed on the mark. The main conclusions that the critics have drawn are also correct: business-to-business e-commerce is expensive and difficult to set up and maintain, because of shortcomings in the design concepts underlying standard EDI.

The problem is particularly acute for SMEs (small and medium-sized enterprises), which all too often find the cost of initial setup prohibitive. This cost includes substantial investment in legal services, management and staff time, and software. In the world of EDI, this is often referred to as the *first trade problem*. Once an EDI system is up and running smoothly, the incremental costs can be quite low and the incremental benefits very high. The problem is getting the *first* messages sent and working properly. That typically involves a large fixed cost.[3]

And, the first trade problem is hardly peculiar to the world of EDI. More broadly, it is recognized that there are important conceptual and technical issues to be addressed if the common vision—of millions of artificial agents cruising the Internet and doing deals for their owners in *ad hoc* and opportunistic ways—is to be realized. This is sometimes called the *spontaneity problem* [Dic00]; it is a more challenging generalization of the EDI first trade problem.

Something needs to be done to address these problems—first trade, spontaneity, etc.—but what?

A central issue—and the focus of this paper—is the fundamental question of *semantic transparency*: When S (speaker) sends A (addressee) a message, how does A's machine (or indeed A itself) know what the message is about, what it means? Or, given that A needs to understand a particular message, how can this be achieved at low cost and hence in a maximally-automated fashion?

Given proper standards, message meanings are determined and computers can be programmed to act appropriately to the intended message meanings. This is something that happens millions of times daily. The rap against EDI has been that proper standards have not in fact been created, and indeed cannot be made because of the misguided way in which the EDI standards are designed. The worry associated with the spontaneity problem is that we

[1] See, e.g., [Emm93,Emm94,Kim91] for valuable general introductions to electronic data interchange (EDI).

[2] For a sample of the criticisms, many from a basically friendly perspective, see: [AY96], [BLW97], [Dic00], [Kim91], [Kim99], [KM93a], [KT00], [Leh96], [Sal95], [Ste94], and [Ste96].

[3] On the EDI first-trade problem see [Lee99,TT98a] and papers throughout [AY96].

do not know how to design a communication regime that will support the vision.

Into this unresolved situation comes XML. Proponents of XML have been touting XML's strengths and claiming that XML overcomes, or can overcome, the semantic transparency problem in general, and the e-commerce first trade problem in particular. It would serve no constructive purpose to rehearse the particular misstatements and exaggerations that have been committed by the advocates of XML. Rather, the question is—regardless of what anyone has claimed—What can XML do to address the transparency problem in e-commerce?[4] The aims of this paper are to answer this question (regarding XML's capabilities), and—since the answer will be a negative one, albeit qualified—to state positively something on how the transparency problem may be solved. To this end, the paper is organized as follows.

In §2 I briefly review why XML (or anything like it) might be–and has been—thought capable of contributing to solving the transparency problem. I assume a passing familiarity with XML on the reader's part. While XML has considerable virtues, I shall argue that it cannot in any very meaningful sense solve the transparency problem. §3 is devoted to the more fundamental discussion of how meaning is communicated between machines (and indeed people). There is nothing irredeemably mysterious here. Once we remind ourselves how it works, we see better why XML cannot solve the transparency problem and we can see better how the problem can be addressed. §§4–5 carry the main burden of the argument. In §4, I provide a formal grammar for microFLBC, a simple but surprisingly powerful language. microFLBC, I shall argue, can be used to provide a genuine semantics for business messaging and can provide a foundation for addressing the transparency problem. In §5, we return to XML and I describe through an example how XML, in conjunction with microFLBC, can yield actual progress on the transparency problem. §6 concludes, and positions the small degree of progress evidenced here in the context of the much larger problem.

Now to the details.

2 XML's Pertinent Virtues and Limitations

HTML's simplicity and beauty of design doubtless contributed enormously to the extraordinarily rapid acceptance of the World Wide Web. Documents marked up in HTML may easily be stored, served from, and read on essentially any widely used computer, using any standard display device. Not only does HTML support a quite usable set of document display features (lists, tables, images, etc.), but it has built-in mechanisms for enabling hypertext linking and email addressing (among many other useful features). Add the fact that the elements of HTML can be learned and applied in just a few

[4] The curious reader, however, can look to [Dri97], [GP98], and [XML98] for examples of honest, perhaps too exuberant, overstatement.

hours, and we can begin to appreciate a truly elegant and effective technical design.

Given all these virtues, what's wrong with HTML and why is there so much excitement about replacing it with XML? Why not, as with other technologies, proceed with incremental refinement? Quite simply, the demand for new features far outstrips the ability of any standards–setting body to keep up. Further, if mutliple standards proliferate the World Wide Web will lose one of its greatest strengths: universal access. Even Microsoft supports XML strongly, thereby eschewing any attempt to dictate a universal standard for HTML.

But despair in keeping up with demand is not the only major source of criticism of HTML. Perhaps even more important is HTML's nearly exclusive orientation towards display of documents. HTML tags—the markup supported by HTML—are nearly entirely content neutral. It is both their main virtue and their main weakness that they apply to all documents, regardless of content, regardless of what the documents are about. A numbered list is the same in a public presentation, a private memo, or an invoice. This facilitates display by arbitrary browsers, but it hinders processing of the underlying document by specialized applications. It is just this latter purpose that must be served for electronic commerce. As envisioned by the many proponents of XML, electronic data interchange (EDI) messages will migrate away from the present standards (e.g., X12 and EDIFACT) and be expressed in XML. But the point of EDI is computer-to-computer exchange of information. The messages must be generated and processed as automatically as possible. This, in turn, requires that the structure of a message substantially informs the processing of the message. A list of items cannot, as in HTML, be *merely* a list; we need to know, e.g., whether it is a list of items to be sold or items to be bought.

XML—as well as SGML, its larger, more general but too unwieldy for the Web parent—has much to offer by way of addressing these two problems. For present purposes, it is fair to say that there are two key moves or ideas present in XML (and SGML). First, the philosophy is to use markup (the tags) for specifying content. Presentation, or display, instructions are specified in a separate, general accompanying document, called the *stylesheet*. This addresses our second problem, above. Documents are marked for processing purposes. Stylesheets, which apply to all documents in a given class and which can be referenced in particular documents, may be used by browsers in presenting the documents on screen or in print.

The second key move addresses our first problem, above, that of too many features and forms. If documents are to be marked for processing purposes, are not those purposes manifold? Indeed they are. How, one then wonders, does this increasing of the uses for marked up documents cohere with the need to reduce the onrush of features in our markup language? Is there something about XML (SGML) that circumvents the need in HTML to add new fea-

tures? Indeed there is: the DTD (document type definition). The purpose of a DTD is to define correct usage of the tags. HTML has a fixed DTD, which needs to be changed whenever new features are added. XML (and before all of this, SGML) has a dynamic DTD mechanism, in the sense that each document can contain (or refer to) its own DTD. There is a fixed specification for valid DTDs, so a document processer need only "know" how to read DTDs in general when it encounters a new XML (SGML) document with a heretofore unseen DTD. The new document has tags that the processor (e.g., browser) has never seen before (or been told of), but the processor can automatically handle the document because it understands DTDish (my name for the DTD language). DTDish is a language language, and it tells the processor about interpreting the tags in the document. Think of DTDish as a language for instructions. Every (XML/SGML) document comes with a set of instructions pertaining to how the document is to be processed. So long as these instructions are in a language understandable by the processor, it will be possible for the processor to handle the document properly, i.e., according to the instructions.

The reason we don't need a standards organization to define what tags there are and what they mean is that this information is provided anew with every document. Anyone can define new tags, or give new definitions to old tags. So long as the definitions are in a valid accompanying DTD, then any processor that understands DTDish will understand the newly defined tags. Thus, XML (and SGML) documents are often said to be "self-describing" [Lig97].

These are lovely ideas. The fact that SGML has been an international standard since 1986, and is widely accepted and used (although not on the Web) attests incontrovertably to its practicality. Nor is there anything importantly special about XML that would make it less useful for electronic commerce than SGML. Quite the contrary. XML is a distilled version of SGML with Web-related additions. XML can be, is being, will be, and probably should be used for EDI purposes in electronic commerce. Even so, it cannot by itself solve the first trade, or transparency, problem in electronic commerce. This, despite widespread intimations and outright claims to the contrary.

To see why XML, nor anything at all like it, cannot solve the transparency problem, we need briefly to consider some fundamentals about communication. With that in hand, I will discuss how, and to what degree, the transparency problem *can* be solved. This in turn will take us back to XML and to an understanding of its proper role in EDI and electronic commerce.

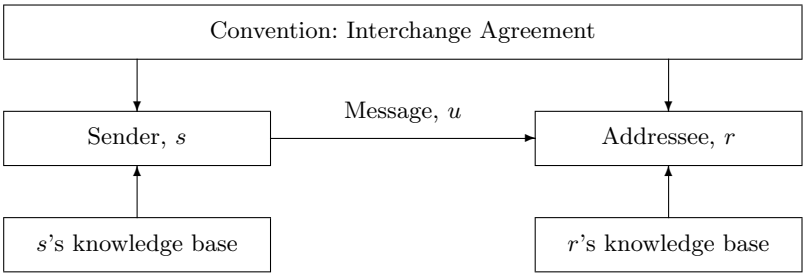


Fig. 1. Basic Messaging Framework: Message u from speaker s to addressee r (after [KT00])

3 Communications Requisites

Consider in the abstract how business messaging—and EDI messaging in particular—is effected. See Figure 1, the basic EDI messaging schema. When two organizations, s and r, wish to do business with EDI they typically begin by negotiating an Interchange Agreement. This is a contract which, among other things, specifies rules of trade between s and r, including which EDI protocols and transaction sets are to be used, and how EDI messages are to be interpreted. Once this contract is in place the sender of a message can draw upon its own knowledge bases, Background Knowledge (e.g., relevant laws and administrative rules), and the Interchange Agreement to formulate a specific message, u (utterance) in the Figure. Similarly, the receiver of a message draws upon its own knowledge bases and the Background Knowledge (including the Interchange Agreement) in order to interpret and act upon the incoming message, u.

All of this is well and good, and even inevitable. The first trade (or transparency) problem is not a complaint against the basic schema. Rather it is a complaint regarding the cost of constructing the Interchange Agreement, and in particular regarding the cost of fixing the meanings of the messages to be exchanged so that business may be conducted automatically. EDI standards are supposed to reduce this cost. Perhaps they do, but the general consensus is that in fact much manual labor, for analysis and negotiation, is required before two parties will have achieved sufficient agreement to do business by machine. Hence the goal: *design a messaging regime that will support common business transactions and that will allow original messages to be composed by machine, transmitted, and understood by the machine of a new trading partner.* The first trade problem is simply our name for the fact that this goal has not been achieved.

Examining and reflecting on some examples of computer-based communication will help us see certain fundamental principles, which in turn will point towards a solution to the first trade problem. Consider, then, a case of primitive (and very common) computer-to-computer communication. Fig-

ure 2, adapted from [McD85], is an example of a 'conversation' between two computers.

1. Computer A: `1, 360`
2. Computer B: `6, 350`
3. Computer A: `3, 1, 1, 2`
4. Computer B: `5, 1, 1`
5. Computer A: `3, 1, 1, 3`
6. Computer B: `5, 1, 1`
7. Computer A: `3, 1, 1, 1`
8. Computer B: `4, 1, 1`
9. Computer A: `3, 2, 1, 150`
10. Computer B: `4, 2, 150`
11. Computer A: `6`
12. Computer A: `8`
13. Computer B: `6`

Fig. 2. Example of a Process–to–Process Dialog

This is an illustration of what in the trade is called *hand-shaking* between devices, here processes running on computers. But what does it mean? Although the meaning is hardly transparent at first, in fact the exchange is easily decoded.[5] Here, in Figure 3, is roughly what this means in English. But how is it that the dialog in Figure 2 comes to have the meaning shown

1. Computer A: Please talk to me on lines 360/361.
2. Computer B: OK. You can talk to me on 350/351.
3. Computer A: Can you do CVSD?
4. Computer B: No, but I can do LPC.
5. Computer A: Can you do RELP?
6. Computer B: No, but I can do LPC.
7. Computer A: How about LPC?
8. Computer B: LPC is fine with me.
9. Computer A: Can you use 150 microsecond sampling?
10. Computer B: I can use 150 microsecond sampling.
11. Computer A: I am ready.
12. Computer A: Are you ready?
13. Computer B: I am ready.

Fig. 3. Example of a Process–to–Process Dialog: Translated into English

in Figure 3? No mystery at all: In one way or another, the owners of the two

[5] See [BK95] for the distinction between decoding and inference.

computers agreed ahead of time what all the messages would mean, and then they programmed their machines to act appropriately. Points arising:

1. First trade
   The communication scheme here could be effected without the two computer owners entering directly into extended discussions regarding what messages there will be and what they will mean. This could happen if someone published a communications protocol adequate for the purposes and it was decided to use that protocol. This would, or could, essentially solve the first trade problem (or its analog in the present case). The solution is rather like what is in fact in place for modem-based communications. Standards are published on how modems are to do handshaking, and manufacturers simply implement to the standards.
   Then why is there a first trade problem in electronic commerce, if there isn't a first handshake problem with Internet service providers?
2. Complex communications
   The source of the problem is complexity. In the current example (or range of examples), only a few things need to be said. We arbitrarily name those things, e.g., with numbers, and we publish a table, as it were, of the required names and what they stand for. This is not possible in electronic commerce. There is no way that at any one time we can come usefully close to identifying—to listing in a table—everything that needs to be said. Even if we were to stick to something as basic as a purchase order, there is literally a combinatorial explosion of firms, trade conditions, prices, quantities, items, and so on. Some other approach, other than that evidenced in Figure 2, is required.

Now the second example. Consider a different but still very simple example of computer-based communication. Without loss of generality, the example uses standard Lisp. We begin by defining a symbol—`m` for message—for an arbitrary arithmetic expression:

```
(setq m '(* 5 (- 6 4)))                    (1)
```

If we ask the Lisp intepreter for the value of `m` (by typing the symbol)

```
m                                          (2)
```

we see returned its value, the arithmetic expression:

```
(* 5 (- 6 4))                              (3)
```

If we now ask Lisp to evaluate `m`

```
(eval m)                                   (4)
```

it correctly calculates the result and returns `10`. This utterly normal, commonplace interaction bears reflection in the present context. Remarking on

something that is usually quite unremarkable will help us understand the first trade problem. Patience will yield insight.

Notice that in this little interaction we have—or by strong analogy nearly have—created a message, sent the message to a computerized process, and the process has correctly interpreted what the message means. Notice further that so long as the message (the symbol `m`) is well formed, Lisp's `eval` will interpret it properly, *even if the message expresses an arithmetic formula that happens never to have been created previously during the history of human kind.* Nothing surprising in this, except to note that in a way we have solved the first trade problem. More generally: the message is semantically transparent to the interpreter. Of course, it was built that way. But how does it work? My claim is that a properly general description of what is going on here must also apply to any regime that hopes to solve the transparency (first trade) problem in electronic commerce.

Understood aright, our little Lisp story contains three essential elements for solving our problem. First, the Lisp `eval` function is able to evaluate (interpret) the message because:

1. The message conforms to a formal grammar, which is used by the `eval` function, and
2. Using the grammar the message is composed ultimately of primitive symbols, which `eval` also knows how to evaluate.

In short, there is a *compositional* formal language for the evaluator to work with. Starting with certain basic elements, a grammar is present that specifies how these elements may be composed into larger expressions. Our basic elements can be specified in a lexicon, using BNF. We need the digits, the arithetic functions, and markers for whitespace.

1. ⟨*digit*⟩ ⟶ `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`
2. ⟨*arith-functor-1*⟩ ⟶ `+` | `-` | `*`
3. ⟨*arith-functor-2*⟩ ⟶ `/`
4. ⟨*S*⟩⟶ `#x20` | `#x9` | `#xD` | `#xA` | ⟨*S*⟩⟨*S*⟩

The gammar composes messages from the lexicon, using its elements as primitives. Again, we use BNF.

1. ⟨*message*⟩ ⟶ ⟨*number*⟩ | (⟨*arith-functor-1*⟩ ⟨*S*⟩ ⟨*message-list*⟩) | (⟨*arith-functor-2*⟩ ⟨*S*⟩ ⟨*message-list*⟩ ⟨*S*⟩ ⟨*message-list*⟩)
2. ⟨*message-list*⟩ ⟶ ⟨*message*⟩ | ⟨*message*⟩ ⟨*S*⟩ ⟨*message-list*⟩
3. ⟨*integer*⟩ ⟶ ⟨*digit*⟩ | ⟨*digit*⟩⟨*integer*⟩
4. ⟨*float*⟩ ⟶ ⟨*integer*⟩ . ⟨*integer*⟩
5. ⟨*number*⟩ ⟶ ⟨*integer*⟩ | -⟨*integer*⟩ | ⟨*float*⟩ | -⟨*float*⟩

Thus, with a total of 18 primitive symbols in the lexicon, the grammar covers an infinite number of arithmetic expressions. Communicants only need prior agreement on the 18 primitive symbols and the 5 rules in the grammar.

One way or another, any genuinely efficient electronic commerce regime of computer-to-computer communications will require this: a fundamental lexicon plus a formal grammar for composing messages. But we need something else. The second essential facet of this example is broadly called a *semantics.* The lexicon and grammar are necessary for any reasonably rich communication regime, but they are not sufficient. They need to *mean* what we want them to mean. The example to hand works fine for arithmetic expressions, but would not work for other kinds of meanings. If we want to talk about baseball, or move funds between bank acounts, or request payment for services, then in spite of their other nice features, our arithmetic lexicon and grammar just won't do.

So, how do we get a formal language (lexicon + grammar) to mean what we want it to mean? We can stipulate meanings for the terms in the lexicon, just as in our first example. The grammar should be understood as a model, a formal structure whose behavior mimics in relevant ways a certain part of the world. As is true for other kinds of models, we can define a grammar however we will, but then we must engage in empirical investigation to determine whether in fact it maps well to that portion of the world we wish to capture. The matching will typically be at best approximate (think of models in the management sciences). The question of whether the model is sufficiently accurate will typically be resolved only by careful study and judicious decision. In the present example, the decision of whether the language properly models arithmetic expressions is a straightforward one. The larger question, of what a language would have to be to properly model what needs to be said in electronic commerce, is of course much more difficult and not yet resolved. (But I shall have something to say about that in the next sections.)

Finally, there is a third important facet of the current example that is relevant to the discussion of communication for electronic commerce. The grammar more or less completely spans its intended domain, in that every valid arithmetic expression is either recognized by our language, or (if it uses a different syntax) can be automatically transformed into an equivalent expression that is recognized by the language. A language to span all of electronic commerce is a feckless goal. But the other extreme—utterly ad hoc languages, special-built for each application—is a large part of why the first trade problem now looms so large.[6]

With these principles of communication in mind, let us see—if only by glimpse—how we might find our way to a workable solution to the first trade, or transparency, problem in electronic commerce.

[6] See [KM93b] for a detailed discussion of the spanning problem in electronic commerce.

4 microFLBC and the Transparency Problem

For some time, I have been engaged in the problem of designing a general-purpose formal language for business communication (FLBC).[7] To date, a main emphasis of this work has been on the challenge of modeling speech acts, such as promising, requesting, accepting, asserting, and so on. Speech acts are, I believe, central to business communication. A purchase order is a request for payment. An offer is a promise, conditioned on acceptance of the offer. A contract is an exchange of promises, among other things. And so on. Systematic examination of EDI messages reveals that in fact they are closely organized around speech acts.[8] Yet, how to formalize speech acts has been an open question. I have developed the elements of such a theory—based on what I call *lean event semantics*—and have presented it elsewhere (see references above). Recounting the lean event theory is not my aim here. Instead, my aim is to *exploit* the theory for the purpose of making progress on the first trade problem.[9]

Continuing the line of argument from the previous section, I offer here (for the first time[10]) a precisely defined, albeit small, language (lexicon + grammar) for electronic commerce. The purpose to hand, recall, is to indicate in a fundamental way how the first trade (or semantic transparency) problem can be satisfactorily resolved. Much work remains to be done and this is not a draft standard for the ISO.

The language to be defined here is microFLBC. It is a fragment of first-order modal logic. I assume a standard formulation of the underlying logic and make no particular assumption regarding which modal system is in play. I have S5 in mind, but the reader is free to pick a favorite system.

Here is the grammar for microFLBC, version 1.0:

1. ⟨*microFLBC-utterance*⟩ ⟶ (⟨*i-force*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ □ ⟨*i-content*⟩)
2. ⟨*i-force*⟩ ⟶ (⟨*speech-act-predicate*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*speaker-predicate*⟩) | (⟨*i-force*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*i-force-thematic-role-predicate*⟩)
3. ⟨*speech-act-predicate*⟩ ⟶ ⟨*speech-act-verb*⟩ (⟨*eventuality-ref* ⟩)
4. ⟨*speaker-predicate*⟩ ⟶ *Speaker* (⟨*eventuality-ref* ⟩ , ⟨*S*⟩ ⟨*general-ref* ⟩)

[7] See, e.g., [Kim90], [Kim99], [KL86], [KM97], [KT00], and references cited therein.

[8] See [Kim98a], [KM93b], [Moo93], [Moo98], [Moo00a], [Moo01], [Sin93], and [Sin98].

[9] Lean event semantics is an extension of what is now called ESΘ theory: event semantics with thematic rôles (aka: subatomic semantics). See [Par90,LS95]; also [Kim98a,Kim97].

[10] Previously, e.g., [KM97], Moore and I have presented a formal language for business communication. That language, however, is what I now call an *application FLBC*. Here, the language being defined aims at capturing the fundamental semantics for the application, and I call it a *semantic FLBC*. The relationship between the two will be clarified in the following section when we return to XML.

5. ⟨*i-force-thematic-role-predicate*⟩ ⟶ ⟨*i-force-thematic-role*⟩ (⟨*eventuality-ref* ⟩ , ⟨*S*⟩ ⟨*general-ref* ⟩) | (⟨*i-force-thematic-role-predicate*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*i-force-thematic-role-predicate*⟩)
6. ⟨*i-content*⟩ ⟶ (⟨*condition*⟩ ⟨*S*⟩ → ⟨*S*⟩ (⟨*speech-act-auxiliary-predicate*⟩ ⟨*S*⟩ ↔ ⟨*S*⟩ ⟨*content*⟩))
7. ⟨*condition*⟩ ⟶ ⊤ | ⟨*speech-act-auxiliary-predicate*⟩
8. ⟨*speech-act-auxiliary-predicate*⟩ ⟶ ⟨*speech-act-auxiliary*⟩ (⟨*eventuality-ref* ⟩)
9. ⟨*content*⟩ ⟶ ⟨*microFLBC-utterance*⟩ | ⟨*simple-content*⟩ | (⟨*content*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*content*⟩)
10. ⟨*simple-content*⟩ ⟶ (⟨*simple-content-verb-predicate*⟩) | (⟨*simple-content-verb-predicate*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*content-predicates*⟩)
11. ⟨*simple-content-verb-predicate*⟩ ⟶ ⟨*ordinary-verb-predicate*⟩ | ⟨*speech-act-auxiliary-predicate*⟩
12. ⟨*content-predicates*⟩ ⟶ ⟨*content-predicte*⟩ | (⟨*content-predicte*⟩ ⟨*S*⟩ ∧ ⟨*S*⟩ ⟨*content-predicte*⟩)
13. ⟨*content-predicte*⟩ ⟶ ⟨*thematic-role-predicate*⟩ | ⟨*prepositional-predicate*⟩ | ⟨*misc-predicate*⟩ | ⟨*domain-specific-predicate*⟩
14. ⟨*ordinary-verb-predicate*⟩ ⟶ ⟨*ordinary-verb*⟩ ⟨*eventuality-ref* ⟩
15. ⟨*thematic-role-predicate*⟩ ⟶ ⟨*thematic-role*⟩ ⟨*event-arg-pair*⟩
16. ⟨*event-arg-pair*⟩ ⟶ (⟨*eventuality-ref* ⟩ , ⟨*S*⟩ ⟨*general-ref* ⟩)
17. ⟨*prepositional-predicate*⟩ ⟶ ⟨*preposition*⟩ ⟨*event-arg-pair*⟩
18. ⟨*misc-predicate*⟩ ⟶ ⟨*misc1*⟩ ⟨*general-ref*⟩ | ⟨*misc2*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩) | ⟨*misc3*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩ , ⟨*general-ref*⟩)
19. ⟨*domain-specific-predicate*⟩ ⟶ ⟨*domain-specific1*⟩ ⟨*general-ref*⟩ | ⟨*domain-specific2*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩) | ⟨*domain-specific3*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩ , ⟨*general-ref*⟩)
20. ⟨*general-ref* ⟩ ⟶ PCDATA | ⟨*function1*⟩ ⟨*general-ref*⟩ | ⟨*function2*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩) | ⟨*function3*⟩ (⟨*general-ref*⟩ , ⟨*general-ref*⟩ , ⟨*general-ref*⟩)
21. ⟨*eventuality-ref* ⟩ ⟶ ⟨*general-ref* ⟩

Lexicon 1.0 for microFLBC. Terminals/lexicon for microFLBC:

1. ⟨*S*⟩⟶ `#x20` | `#x9` | `#xD` | `#xA` | ⟨*S*⟩⟨*S*⟩
2. ⟨*speech-act-verb*⟩ ⟶ *promise* | *assert* | *declare* | *request* | *purchase-order* | *cancel* | *confirm* | *endorse* | *invoice* | *inquire* | *estimate* | *order* | *report* | *specify* | *clear* | *approve* | *void* | *discharge*
3. ⟨*ordinary-verb*⟩ ⟶ *pay* | *deliver* | *ship* | *arrive* | *become-due* | *become-effective* | *change* | *examine* | *count* | *create* | *expire* | *fail* | *manufacture* | *open* | *close* | *perform* | *place* | *process* | *arrive* | *depart* | *load* | *unload* | *receive*
4. ⟨*i-force-thematic-role*⟩ ⟶ *Addressee* | *Theme* | *Cul* | *Comc* | *Hold* | *Sake*
5. ⟨*speech-act-auxiliary*⟩ ⟶ *V* | *K* | *H* | *Auth*
6. ⟨*thematic-role*⟩ ⟶ *Agent* | *Performer* | *Experiencer* | *Benefactive* | *Goal* | *Theme* | *Sake* | *Cul* | *Comc* | *Hold* | *Location* | *Source* | *Instrument*

7. ⟨*preposition*⟩ ⟶ *to* | *from* | *into* | *at* | *for* | *with* | *by* | *in*
8. ⟨*misc1*⟩ ⟶ *red* | *green* | *blue*
9. ⟨*misc2*⟩ ⟶ ≤ | = | < | > | ≥ | *description* | *during* |
10. ⟨*misc3*⟩ ⟶ *unit* | *quantity* | *unitprice*
11. ⟨*function1*⟩ ⟶ *day*
12. ⟨*function2*⟩ ⟶ `+` | `-` | `*` | `/`
13. ⟨*domain-specific2*⟩ ⟶ *FOB* | *TERMS*

4.1 Representing a Simple Promise

Our first example is a simple promise, of the sort that is made many times daily in the conduct of commerce.

At time t, s promises r that s will deliver to r goods g within 30 days.

(5)

Using an EDI-like message notation, this might be expressed formally (but not logically) as:

```
1. promise : 12345
2. date-time : 1999-09-23
3. from : s
4. to : r
5. deliver
   (a) goods : g
   (b) to : r
   (c) by : s
   (d) date-time : 1999-09-23 + day(1999-09-23 + 30)
```

(6)

So, we should think of expression (6) as an instance of u in Figure 1.

Using my lean event sementics [Kim99] and microFLBC, expression (5) is rendered into first-order modal logic as follows:

$$promise(12345) \wedge Speaker(12345, s) \wedge Addressee(12345, r) \wedge Cul(12345, 1999\text{-}09\text{-}23) \wedge \Box(\top \rightarrow (K(12345) \leftrightarrow (deliver(e) \wedge Agent(e, s) \wedge Benefactive(e, r) \wedge Sake(e, 12345) \wedge Theme(e, g) \wedge Cul(e, t) \wedge \leq(t, +(1999\text{-}09\text{-}23, day(1999\text{-}09\text{-}23 + 30))))))$$

(7)

Comments and points arising:

1. Expression (7) reads roughly as "12345 is a promise by s to r, occurring on September 23, 1999. The promise 12345 concerns a delivery event e. Necessarily, this promise is kept if and only s effects a delivery to r of g, and this is for the sake of the promise 12345. In addition, the delivery, e must occur (for the promise 12345 to be kept) within thirty days of September 23, 1999."
2. This reading is also (I claim) a correct intepretation of the EDI-like message, Expression (6) and of the original ordinary language-like utterance, Expression (5).
3. $\Box$ in Expression 7 is a modal logic necessity operator. For any sentence, P, read $\Box P$ as "Necessarily, P" or "It is necessary that P." This necessity operator is required for technical reasons, which for present purposes I elide. It is generally safe simply to drop the operator, $\Box$. Doing this yields the *extensional approximation,* and for practical purposes the approximation suffices. Sticking to principles, the grammar for microFLBC retains the requirement that the $\Box$ be present.
4. Unlike Expression (6), Expression (7) is logical and fully formal. Note in this regard how Expression (7) makes explicit so much that would otherwise have to be read into or assumed regarding Expressions (5) and (6), which are open to more than one interpretation.
5. The predicates in Expression (7) all occur in the lexicon, and are of general utility. They could be used in representing very many types of primary messages.
6. There is nothing special about promising that limits the scope of this approach to representing messages. The move in evidence here generalizes to the other illocutionary forces, such as asserting, requesting, and declaring. Although further details are required, the analysis is remarkably simple. Even so, it is able to capture the core logical behavior of various illocutionary forces.[11]
7. Two kinds of semantic theory are present: a theory applying to natural language and a theory applying to first-order modal logic. Expression (7) exploits lean event semantics, a partial theory of natural language semantics, to represent a meaning in logic. Lean event semantics extends ESΘ theory [Par90,LS95]. See also [Kim98a,Kim97]. I am assuming (and have argued elsewhere) that this theory provides us with an adequate account of the semantics of the sorts of expressions exhibited for discussion here (i.e., messages in electronic commerce). If it does, then—I am now arguing—we can see a way through to solving the first trade problem in electronic commerce. The second semantic theory to hand is the semantics for first-order modal logic. This theory is well-established and I take it to be unproblematic.

[11] See [Kim90], [Kim97], [Kim98a], [Kim98b], and [Kim99].

8. $\top$, appearing in item (7) of the microFLBC grammar, is shorthand for any logical tautology. It always evaluates to true. Anything conditioned on $\top$ is thus only vacuously conditioned. If I say I will go to the store whether or not it rains, my going is but vacuously conditioned on the weather.
9. PCDATA, appearing in item (20) of the microFLBC grammar, alludes to the XML use of that expression: basically PCDATA matches any string of characters. This is where we give up specifying structure and content ourselves with names of things.
10. microFLBC contains very little by way of validation principles. Data types are not recognized and no real distinction is made between names of eventualities (events, processes, and states) and names of other kinds of things (e.g., people and products). Every utterance act must have a *Speaker*, but little else is required. These are important matters indeed, but they are beyond the scope of this paper.

Now to a more complex example.

4.2 Representation of a Simple Purchase Order

Figures 4 and 5 show, respectively, a simple example of a specific purchase order and a representation of it in an EDI-like (structured, not logical) syntax. I have kept the names of data items short, and the number of data items few, for the sake of convenience. The example is rich enough to bear the points I wish to make, and sparse enough to avoid burdening the reader with unnecessary detail. Actual purchase orders, and the EDI protocols for them, are of course much more complex. Full treatment of such examples is beyond the scope and purposes of this paper.

Figure 6, page 218, displays the results of using lean event semantics and microFLBC to represent the purchase order of Figures 4 and 5. Although there is very much to say about this example, I shall give just a few remarks, aimed at making the example accessible and its connection with the main points clear.

1. The expression in Figure 6 is a fully formal, logical *model* (and hence, approximation) of what I think would typically be meant by the purchase order shown in Figure 4.
2. As in the previous example:
   (a) Predicates beginning with a capital letter (*Speaker, Theme, Comc,* etc.) are generic reserved words in lean event semantics;
   (b) Predicates in lower case (e.g., *deliver, pay*) are (here) verbs whose meaning is specified independently (more on this below); and
   (c) Predicates in all upper case (e.g., *FOB, TERMS*) are domain-specific.
   All of these predicates appear in the microFLBC lexicon.
3. The expression in Figure 6 is a single logical conjunction with four main blocks of code.

TO: r		CUSTOMER NO. s	
		TERMS 30-days-net	
SHIP TO: an-address		SALES	
		SHIP WEEK OF 1999-12-03	
		FOB customer	
ORDER NO. 54321	DELIVERY VIA	ROUTING	

PLEASE SHIP THE FOLLOWING AS SPECIFIED

ITEM	QUANTITY ORDERED	DESCRIPTION	UNIT COUNT	UNIT PRICE	TOTAL AMOUNT
catid-32-9	box	copier paper	6	$45.01	$270.06
catid-35-9	dozen	no. 2 pencil	3	$1.99	$5.97

SPECIAL INSTRUCTIONS:	TOTAL AMOUNT	$276.03
	DATE 1999-11-19	APPROVAL SIGNATURE
	PURCHASER SIGNATURE	
	TITLE	

***** PURCHASE ORDER *****

Fig. 4. Simple Purchase Order Example

```
 1. purchase-order : 54321
 2. date : 1999-11-19
 3. from : s
 4. to : r
 5. ship-to: an-address
 6. terms: 30-days-net
 7. ship-week-of: 1999-12-03
 8. FOB: customer
 9. grand-total: 276.03
10. deliver
    (a) goods : catid-32-9
    (b) unit: box
    (c) description: "copier paper"
    (d) quantity: 6
    (e) unit-price: 45.01
    (f) line-total: 270.06
11. deliver
    (a) goods : catid-35-9
    (b) unit: dozen
    (c) description: "no. 2 pencil"
    (d) quantity: 3
    (e) unit-price: 1.99
    (f) line-total: 5.97
```

Fig. 5. EDI-like representation of the simple purchase order in Figure 4

4. The gist of the first block is that this is a purchase order from s to r happening ("culminating," *Cul*) on 1999-11-19.
5. The gist of the second and third blocks is to indicate that this purchase order is making a request (of r by s). The request is honored (H) if and only if two shipping events occur, *e1* and *e2*. Descriptions of these events follow and match to the original purchase order. Note that in both cases, the shipping events are (requested to) begin ("commencing", *Comc*) during the period indicated. Nothing is said about when delivery should occur. Also, both both events are vacuously conditioned on $\top$, a generic symbol for something that is always true. Vacuous conditioning is no conditioning at all.
6. The gist of the fourth block is that if the afore-described request is in fact honored (here the conditioning is nonvacuous), then this purchase order is also making a promise. The promise is kept (K) if and only if *e3* is a paying event with the properties described.
7. Note that the expression is fully logical and supports inferencing. For example, suppose the purchase order is issued and in fact r honors it fully (ships the goods as, when, and where described). Suppose further that s fails pay r as described. It follows logically (as it should) that the

purchase-order(54321) ∧ Speaker(54321, s) ∧ Addressee(54321, r) ∧ Cul(54321, 1999-11-19) ∧

$\Box$*((*$\top$ *→ (H(54321)* $\leftrightarrow$

((ship(e1) ∧ Agent(e1, r) ∧ Benefactive(e1, s) ∧ to(e1, an-address) ∧ Theme(e1, catid-32-9) ∧ Sake(e1, 54321) ∧ unit(e1, catid-32-9, box) ∧ description(catid-32-9, "copier paper") ∧ quantity(e1, catid-32-9, 6) ∧ Comc(e1, t1) ∧ FOB(e1, customer) ∧ during(t1, week-of(1999-12-03))) ∧

(ship(e2) ∧ Agent(e2, r) ∧ Benefactive(e2, s) ∧ to(e2, an-address) ∧ Theme(e2, catid-35-9) ∧ Sake(e2, 54321) ∧ unit(e2, catid-35-9, dozen) ∧ description(catid-35-9, "no. 2 pencil") ∧ quantity(e2, catid-35-9, 3) ∧ Comc(e2, t2) ∧ FOB(e2, customer) ∧ during(t2, week-of(1999-12-03)))))) ∧

(H(54321) → (K(54321) $\leftrightarrow$

*(pay(e3) ∧ Agent(e3, s) ∧ Benefactive(e3, r) ∧ Sake(e3, [e1, e2]) ∧ Theme(e3, p3) ∧ unit(e3, p3, $) ∧ quantity(e3, p3, q) ∧ unitprice(e1, catid-32-9, p1) ∧ unitprice(e2, catid-35-9, p2) ∧ unit(e1, p1, $) ∧ unit(e2, p2, $) ∧ quantity(e1, p1, 45.01) ∧ quantity(e2, p3, 1.99) ∧ =(p3, +(*(45.01, 6), *(1.99, 3))) ∧ Cul(e3, t3) ∧ TERMS(e3, 30-days-net)))))*

Fig. 6. Representation, via lean event semantics, of a simple purchase order, Figures 4 and 5

promise associated with the purchase order, made by s, has not been kept, i.e., that $\neg K(54321)$.

4.3 Mapping to the World

No semantics can ever be entirely formal. At some point we need to map between the formal symbols used in the formal language, and the system that language is to model. The trick is to give this mapping at a very basic level, and then rely as much as possible on composition of symbols to provide additional meaning. With our logical semantics (e.g., Figure 6) we need to provide a basic mapping for: terms (*s, r,* etc.), functions (e.g., *week-of*), and predicates (e.g., *ship, pay*). Terms are rather straightforward: one makes a catalog, a sort of table, in which the referring expression is mapped to what it refers to. For example, s might be the customer number of the firm Nadir, Inc. Functions are also straightforward: one either maps them, much as terms are mapped, or one defines them by composition from more primitive functions.

Predicates, as noted, are of three kinds:

1. Specific for lean event semantics, e.g., *Theme, Comc, Unit*
2. Generic, e.g., *ship, pay*
3. Domain specific, e.g., *FOB, TERMS*

The specific predicates should be few in number and can be defined (mapped) explicitly. The generic predicates can and should be mapped to a clear, broadly-accepted, and public specification. Language having the importance it does, such specifications are indeed available. WordNet, an electronic lexicon produced over a number of years at Princeton University, is an excellent example [Fel98]. Let us see, briefly, what WordNet has to say about the three generic (verb) predicates in Figure 6: *purchase-order, ship, pay.*

WordNet recognizes *purchase order* as a noun, but not as verb. Nonetheless, what it says is useful. The following passage, as well as all the subsequently quoted passages, is from WordNet 1.6.

```
The noun purchase order has 1 sense
        (no senses from tagged texts)
1. {04902219} <noun.communication> order1#7,
         purchase order#1 --
(a commercial document used to request someone to supply
something in return for payment; "IBM received an order for
a hundred computers")
```

Notice that a purchase order here—and in Figure 6—involves both a request for something and a (promised) payment for it.

WordNet 1.6 recognizes *ship* as a verb for which there is exactly one sense: to transport commercially.

```
1 sense of ship
Sense 1
{01328437} <verb.motion> transport1#4, send#4, ship#1 --
(transport commercially)
   => {01328337} <verb.motion> barge1#2 --
        (transport by barge on a body of water)
   => {01331285} <verb.motion> dispatch#1, despatch#1,
        send off#1 -- (send off promptly)
   => {01331167} <verb.motion> bundle off#1 --
        (send off unceremoniously)
   => {01331450} <verb.motion> route2#1 --
        (send documents or materials to appropriate
         destinations)
   => {01331576} <verb.motion> forward#1, send on#1 --
        (send or ship onward from an intermediate post or
         station in transit; "forward my mail")
```

More refined meanings are available with different verbs. We may thus tentatively identify the *ship* of Figure 6 with *ship#1* of WordNet.

Finally, WordNet 1.6 recognizes 11 distinct meanings for *pay* as a verb.

```
The verb pay has 11 senses (first 11 from tagged texts)
1. {01540968} <verb.possession> pay#1 --
    (give money in exchange for goods or services;
    "I paid four dollars for this sandwich";
    "Pay the waitress, please")
2. {00718708} <verb.communication> give1#5, pay#2 --
    (convey, as of a compliment, regards, attention, etc.;
    bestow; "Don't pay him any mind"; "give the orders";
    "Give him my best regards"; "pay attention")
3. {01541614} <verb.possession> pay up#1, ante up#1, pay4#3 --
    (cancel or discharge a debt; "pay up, please!")
4. {01542031} <verb.possession> pay2#4, pay off4#4, make up#3,
    compensate2#4 -- (do or give something to somebody in
    return; "Does she pay you for the work you are doing?")
5. {01695538} <verb.social> pay#5 --
    (render; "pay a visit"; "pay a call")
6. {00500874} <verb.cognition> pay3#6 --
    (bear (a cost or penalty), in recompense for some action;
    "You'll pay for this!"; "She had to pay the penalty for
    speaking out rashly"; "You'll pay for this opinion later")
7. {01566906} <verb.possession> yield#10, pay1#7, bear1#8 --
    (bring in; as of investments; "interest-bearing accounts";
    "How much does this savings certificate pay annually?")
8. {01869530} <verb.stative> pay#8 --
    (be worth it; "It pays to go through the trouble")
9. {00496485} <verb.cognition> give2#10, pay#9, devote#2 --
    (as in the expressions "give thought to";
    "give priority to", etc.)
10. {01541783} <verb.possession> pay14#10 --
    (discharge or settle; "pay a debt"; "pay an obligation")
11. {01600647} <verb.possession> pay13#11 --
    (make a compensation for; "a favor that cannot
    be paid back")
```

Pretty straightforwardly, we can tentatively identify microFLBC's *pay* with WordNet's *pay#1*.

If our lexicon is to be properly completed, this sort of mapping must be carried through systematically. Further excursions in that direction, however, would only divert us from our main subject.

5 Back to XML

It will help to summarize before returning to discussion of XML. Through a series of examples I have observed that effective computer-to-computer (automated) communication requires:

1. A *common* lexicon, which identifies the primitive expressions and what they mean
   This is what makes possible the simple sort of communication illustrated, above, in Figure 2. If we are content with such an elementary form of communication, then the transparency problem can (in principle) be solved simply by making available to all parties the common lexicon. Each may then program its own computers without consulting the other, and (in principle) the first exchange and all subsequent exchanges will work properly.
   The problem is that for most purposes, especially for electronic commerce, a "words only"—or *atomic*—form of exchange is insufficiently rich, and we must resort to messages *composed* from items in the lexicon.
2. A *common* grammar that specifies the permitted compositional expressions
   Lacking this, it would border on miraculous if the sender composed messages that the receiver could recognize correctly.
3. A *common* convention for interpreting what the expressions produced by the common grammar mean
   This is also known as a semantics for the (common) language. In the two relevant examples above, the Lisp arithmetic expressions and microFLBC, I alluded to but did not specify the associated semantics. For the case of the Lisp expressions, it is the semantics that tells us that applying the functor + to a list of numbers produces their sum, instead of something else. microFLBC, being a fragment of first-order modal logic, inherits the semantics for that language. This is what tells us that $\wedge$ means (roughly, *and* but a truth table can be given to make this rigorous), and indeed what each possible expression in the grammar (BNF) means.[12]

[12] Regarding *common*,, I oversimplify for the sake of the issues at hand. I do not want to suggest that communicants must be looking at exacting the same lexicon, etc. They could have in common different copies of things. Also, they might come to acquire their copies by incremental learning. Skyrms [Sky96, chapter 5] is particularly convincing on this subject, but we are far from seeing our way to applications via learning of meaning.

With these points to hand, we can diagnose the sources of the transparency (spontaneity, or first-trade) problem in EDI as:

1. Lack of a clear and complete lexicon
   Nothing resembling even the above mapping of a few predicates to WordNet is generally available, although in certain domains much has been achieved in this direction.
2. No fully-specified grammar
   The various EDI standards (X12, EDIFACT, etc.) attempt to define compositional expressions, and hence can be seen as providing grammars for business communication. Two major shortcomings are present. First, the grammars are incomplete; they do not fully and rigorously specify the permitted messages. This is often an important locus of negotiation for organizations preparing for their first electronic trade. Second, grammars have proliferated. The standards each identify scores of message types (invoice, purchase order, request for quotation, etc.), called *transaction sets.* Individual 'grammars' (speaking loosely) are defined for each transaction set. This, too, only increases the pre-trade clarification burden.
3. No semantics
   Or nearly no semantics. The standards provide rough and intuitive—and occasionally quite specific—information on what the various fields are for and what can go in them, but this is a very long way from having a full and rigorous semantics. Even if the trading parties are determined to follow the standards exactly, this is impossible because the standards are not exact. Negotiation ensues.

What of XML? It indeed brings much to our table. First, the DTD mechanism offers the prospect of a fully rigorous grammar for business messaging. What a DTD really is is an executable BNF specification. At least in principle (and surely for very many practical purposes), this answers well to the requirement for a formal grammar. In this regard alone, XML presents the prospect of a great improvement over current EDI specification regimes. Second, XML has a namespace mechanism by which it can fix the references of the terms it uses. For example, a catalog number for a product need not be universally unique. If it is locally unique—within a given catalog—and we have a way to refer to the catalog in question, then the product number can (in principle) be used without ambiguity. Exploring this aspect of XML in detail is beyond the scope of this paper, but it is safe to say that the namespace mechanism is at least a promising vehicle for solving the lexicon requirement.

This leaves the question of semantics. Just as a BNF provides the grammar or syntax for a language, and cannot provide the semantics, so XML's DTD mechanism cannot provide a semantics either. As with namespaces, an XML document can point to a semantic specification, but it cannot provide one for itself. What to do? We can give an XML document a semantics by mapping its DTD to another language which itself is or has a formal semantics. What is special (although not unique) about logic is that it can serve

as such a language. Semantics for logic is well worked out, and arguably the microFLBC fragment can properly express much of what we need to say for business messaging. So, the trick is to define XML DTDs that can be mapped to microFLBC. Then, microFLBC (or something like it) and the mapping correspondence with the DTD can together serve as the common semantics. This in principle completes our three requirements for effective communication and—in principle—lets us solve the transparency problem.

I will illustrate with a simple example. Suppose s wants to send a message to a requesting payment for items delivered in response to a previous work order, number 789, originally issued by a. In English the message is:

s requests of a that a pay s for item 789.

In microFLBC this is more carefully stated as

$$request(2345) \wedge Speaker(2345,\ s) \wedge Addressee(2345,\ a) \wedge \Box\ (\top \rightarrow (H(2345) \leftrightarrow (pay(23456) \wedge Agent(23456,\ a) \wedge Benefactive(23456,\ s) \wedge Sake(23456,\ 789))))$$

Here is one way to say this in XML:

```
<microFLBC-utterance>
<i-force>
  <speech-act-predicate>
    <speech-act-verb>request</speech-act-verb>
    <eventuality-ref>2345</eventuality-ref>
  </speech-act-predicate>
  <speaker-predicate>
    <eventuality-ref>2345</eventuality-ref>
    <general-ref>s</general-ref>
  </speaker-predicate>
  <i-force-thematic-role-predicate>
    <i-force-thematic-role>Addressee</i-force-thematic-role>
    <eventuality-ref>2345</eventuality-ref>
    <general-ref>r</general-ref>
  </i-force-thematic-role-predicate>
<i-force>
<i-content>
  <condition>True</condition>
  <speech-act-auxiliary-predicate>
    <speech-act-auxiliary>H</speech-act-auxiliary>
    <eventuality-ref>2345</eventuality-ref>
  </speech-act-auxiliary-predicate>
  <content>
  <simple-content>
    <simple-content-verb-predicate>
     <ordinary-verb-predicate>pay</ordinary-verb-predicate>
```

```
        <eventuality-ref>23456</eventuality-ref>
       </simple-content-verb-predicate>
       <content-predicates>
        <content-predicate>
         <thematic-role-predicate>
          <thematic-role>Agent</thematic-role>
          <event-arg-pair>
           <eventuality-ref>23456</eventuality-ref>
           <general-ref>a</general-ref>
          </event-arg-pair>
         </thematic-role-predicate>
        </content-predicate>
        <content-predicate>
         <thematic-role-predicate>
          <thematic-role>Benefactive</thematic-role>
          <event-arg-pair>
           <eventuality-ref>23456</eventuality-ref>
           <general-ref>s</general-ref>
          </event-arg-pair>
         </thematic-role-predicate>
        </content-predicate>
        <content-predicate>
         <thematic-role-predicate>
          <thematic-role>Sake</thematic-role>
          <event-arg-pair>
           <eventuality-ref>23456</eventuality-ref>
           <general-ref>789</general-ref>
          </event-arg-pair>
         </thematic-role-predicate>
        </content-predicate>
       </content-predicates>
     </simple-content>
     </content>
   </i-content>
   </microFLBC-utterance>
```

There is a more or less transparent mapping between this XML expression and microFLBC. I leave it to the reader as an exercise.

6 Conclusion

If communicating automated agents are to benefit from drastically reduced first-trade costs, it will be necessary to provide them (and their human masters) with:

1. a fundamental semantic language for modeling communication, including
   (a) a lexicon
   (b) a grammar
   (c) a semantics
2. an application language, for business communication,
3. a mapping between the two languages

and all of this must be fully formal and processable by machine. This is a tall order and very much remains to be done. Fortunately, what we have but glimpsed here can be realized in increments. How much would microFLBC have to expanded in order to serve as the semantic foundation for some domain of application, heretofore poorly served by EDI? The prospects, I think, are quite good, but that is something for another paper.

References

[AY96] Nabil R. Adam and Yelena Yesha (eds.), *Electronic commerce: Current research issues and applications*, Lecture Notes in Computer Science, vol. 1028, Springer, Berlin, Germany, 1996.

[BK95] Hemant K. Bhargava and Steven O. Kimbrough, *On embedded languages, meta-level reasoning and computer-aided modeling*, The Impact of Emerging Technologies on Computer Science and Operations Research (Stephen G. Nash and Ariela Sofer, eds.), Kluwer Academic Publishers, Boston, MA, 1995, ISBN 0-7923-9542-5. File: csts-94-meta-sok-hkb., pp. 27–44.

[BLW97] Roger W.H. Bons, Ronald M. Lee, and Renée Wagenaar, *Computer-aided auditing of inter-organizational trade procedures*, Intelligent Systems in Accounting, Finance and Management **6** (1997), no. 2, 29–46.

[Dic00] Kevin Dick, *XML: A manager's guide*, Addison-Wesley, Reading, Massachusetts, 2000, ISBN: 0-201-43335-4.

[Dri97] Chris Driscoll, *XML touted as cure for EDI ills: New markup language extends web capabilities beyond HTML's limits*, Web page, 5 August 1997, `http://www.geocities.com/WallStreet/Floor/5815/edinews01.htm`, accessed 1999-12-28.

[Emm93] Margaret A. Emmelhainz, *EDI: A total management guide*, second ed., Van Nostrand Reinhold, New York, NY, 1993, ISBN: 0-442-312690-9.

[Emm94] ———, *Electronic data interchange in logistics*, The Logistics Handbook (James F. Robeson and William C. Copacino, eds.), The Free Press, New York, NY, 1994, ISBN: 0-02-926595-9., pp. 737–756.

[Fel98] Christiane Fellbaum (ed.), *Wordnet: An electronic lexical database*, The MIT Press, Cambridge, MA, 1998, ISBN: 0-262-06197-X.

[GP98] Charles F. Goldfarb and Paul Prescod, *The xml handbook*, Prentice Hall PTR, Upper Saddle River, NJ, 1998, ISBN: 0-13-081152-1.

[Kim99] Steven O. Kimbrough, *Formal language for business communication: Sketch of a basic theory*, International Journal of Electronic Commerce **3** (Winter 1998–99), no. 2, 23–44.

[Kim90] ———, *On representation schemes for promising electronically*, Decision Support Systems **6** (1990), no. 2, 99–121.

[Kim91] Paul Kimberley, *Electronic data interchange*, McGraw-Hill, Inc., New York, New York, 1991.

[Kim97] Steven O. Kimbrough, *On electronic commerce, subatomic semantics and Cæsar's stabbing*, Proceedings of the Thirtieth Hawaii International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1997, pp. 361–370.

[Kim98a] ———, *On ESΘ theory and the logic of the X12 date/time qualifiers*, Proceedings of the Thirty-First Hawai'i International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1998, pp. 330–339.

[Kim98b] ———, *Sketch of a basic theory for a formal language for business communication*, Proceedings of the Thirty-First Hawai'i International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1998, pp. 717–725.

[KL86] Steven O. Kimbrough and Ronald M. Lee, *On illocutionary logic as a telecommunications language*, Proceedings of the International Conference on Information Systems (San Diego, CA), December 1986, pp. 15–25.

[KM93a] Steven O. Kimbrough and Scott A. Moore, *On obligation, time, and defeasibility in systems for electronic commerce*, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III, Information Systems: DSS/Knowledge-Based Systems (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1993, pp. 493–502.

[KM93b] ———, *On the spanning hypothesis for EDI semantics*, Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, January 1993.

[KM97] ———, *On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity*, ACM Transactions on Information Systems **15** (October 1997), no. 4, 321–367.

[KT00] Steven O. Kimbrough and Yao-Hua Tan, *On lean messaging with unfolding and unwrapping for electronic commerce*, International Journal of Electronic Commerce **5** (2000), no. 1, 83–108.

[Lee99] Ronald M. Lee, *Distributed electronic trade scenarios: Representation, design, prototyping*, International Journal of Electronic Commerce **3** (1999), no. 2, 105–136.

[Leh96] Fritz Lehmann, *Machine-negotiated, ontology-based EDI (electronic data interchange)*, Electronic Commerce: Current Research Issues and Applications (Nabil R. Adam and Yelena Yesha, eds.), Lecture Notes in Computer Science, vol. 1028, Springer, Berlin, Germany, 1996, pp. 27–46.

[Lig97] Richard Light, *Presenting xml*, SAMS Net, Indianapolis, IN, 1997, ISBN: 1-57521-334-6.

[LS95] Richard Larson and Gabriel Segal, *Knowledge of meaning: An introduction to semantic theory*, The MIT Press, Cambridge, Massachusetts, 1995, ISBN: 0-262-62100-2.

[McD85] David McDonald, *Conversations between programs*, Cognitive Constraints on Comunication (Lucia Vaina and Jaakko Hintikka, eds.), Synthese Language Library, vol. 18, D. Reidel Publishing Company, Boston, MA, 1985, ISBN: 90-277-1456-8., pp. 403–424.

[Moo93] Scott A. Moore, *Saying and doing: Uses of formal languages in the conduct of business*, Ph.D. thesis, University of Pennsylvania, The Wharton School, Philadelphia, PA, 19104, USA, December 1993.

[Moo98] ———, *Categorizing automated messages*, Decision Support Systems **22** (1998), no. 3, 213–241.

[Moo00] ———, *KQML and FLBC: Contrasting agent communication languages*, International Journal of Electronic Commerce **5** (2000), no. 1, 109–124.

[Moo01] ———, *A foundation for flexible automated electronic commerce*, Information Systems Research **12** (2001), no. 1, 34–62.

[Par90] Terence Parsons, *Events in the semantics of English: A study in subatomic semantics*, Current Studies in Linguistics, The MIT Press, Cambridge, MA, 1990, ISBN: 0-262-66093-8.

[Sal95] Airi Salminen, *EDIFACT for business computers: Has it succeeded?*, StandardView **3** (March 1995), no. 1, 33–42.

[Sin93] Munidar P. Singh, *A semantics for speech acts*, Annals of Mathematics and Artificial Intelligence **8** (1993), no. I–II, 47–71, Reprinted in [HS98].

[Sin98] Munindar P. Singh, *Agent communication languages: Rethinking the principles*, IEEE Computer **31** (1998), no. 12, 40–47.

[Sky96] Brian Skyrms, *Evolution of the social contract*, Cambridge University Press, Cambridge, UK, 1996.

[Ste94] K. Steel, *Another approach to standardising EDI*, Electronic Markets **12** (1994), 34–47.

[Ste96] Ken Steel, *The standardisation of flexible EDI messages*, Electronic Commerce: Current Research Issues and Challenges (Nabil R. Adam and Yelena Yesha, eds.), Springer-Verlag, Berlin, Germany, 1996, ISBN 3-540-60738-2., pp. 13–26.

[TT98] Yao-Hua Tan and Walter Thoen, *A logical model of transfer obligations in trade contracts*, Accounting, Management and Information Technologies **8** (1998), no. 1, 23–38.

[XML98] The XML/EDI Group, *Home page for the XML/EDI Group*, Web page, January 1998, `http://www.xmledi.net`.

Part II

Applications

Designing Control Mechanisms for Value Exchanges in Network Organisations

Vera Kartseva[1] and Yao-Hua Tan[2]

[1] Free University Amsterdam, Faculty of Economics and Business Administration, Information Systems Group, de Boelelaan 1105, 1081 HV Amsterdam, The Netherlands,
vkartseva@feweb.vu.nl

[2] Free University Amsterdam, Faculty of Economics and Business Administration, Information Systems Group, de Boelelaan 1105, 1081 HV Amsterdam, The Netherlands,
ytan@feweb.vu.nl

Abstract. Contracts and organizational controls to monitor contract compliance are important tools to enhance trust in a fair business transaction in network organisations and electronic commerce in general. In this chapter, we propose a design methodology for such contracts and supporting controls, utilizing inter-organisational value models. We argue that a framework for designing control mechanisms should include three steps: design of an inter-organizational value model, analysis of possible violations of contractual obligations underlying this value model, and design of control mechanisms to detect or prevent such violations. It is shown how the e^3-value methodology, which was developed to design business value models, can be extended to model obligations of parties. We use concepts and ideas from deontic logic (the logic of obligations and permissions) to develop an extension of e^3-value called e^3-value+. The e^3-value+ approach is a design tool for modelling violations of obligations, which can be used in contract drafting and contingency planning for inter-organisational collaboration in network organisations.

1 Introduction

As is well-known, lack of trust is one of the main reasons that companies and consumers do not engage in electronic commerce.[1] Technical means, such as encryption technology and digital signatures to build trust-facilitating services, are available right now. Additionally, current research topics in computer science such as web services and peer-to-peer networking enable providing inter-organisational business processes, which are required for trust enhancing procedures [Bar01,FKT01]. However, advanced technology will never develop trust models without a proper underlying organisation design. For example, although technically speaking an electronic signature is much more reliable than a hand-written signature, the historically proven "paper and

[1] See, e.g., [MCC98,TT98b].

pen" way of signing a contract is more trusted. One can surmise that the reason is because an electronic signature is not backed with a proven infrastructure of legislative practices and underlying control mechanisms that make it possible to solve any disagreement between parties.

In [TT98b] a generic model of trust for electronic was introduced that explains how trust can be created by developing party trust and control trust. In this chapter we focus on control trust. In traditional organisations, control mechanisms, with their main focus on management control, have been studied extensively (see [AG03] and [Ouc79]). For electronic commerce, however, the design of control mechanisms is rather uncharted territory. What is needed is an understanding how to design trustworthy control mechanisms on top of technical possibilities of electronic commerce.

Our definition of *control mechanisms* is as in [TT98b]: procedures and protocols that control and monitor the successful performance of a transaction. Contract negotiation can be seen as a way to ensure legality and protect interests of all parties involved in electronic commerce. Technology-oriented research on electronic contracting[2] typically assumes that a plain textual, natural language representation of the contractual content that can be read and interpreted by business people and lawyers suffices. However, business studies[3] indicate that incomplete contracting practices result in increased opportunism and failure of inter-organisational cooperation. In [DS97,TT01] a need for contract negotiating and drafting methodology and tools is identified. In this chapter we suggest that in electronic commerce incomplete contracting practices have to be solved by the negotiation of a contract as well as its supporting controls. This is a multi-disciplinary task, and it involves obviously legal aspects. Also, computer science issues are relevant (many controls take the form of computer software), as well as inter-organisational business process design (many contracts say how, and in which order, business transactions should be carried out, and by whom). In addition, in electronic commerce a thorough understanding of the corresponding business value model is often lacking [Gor02], which makes contingency planning of contracts more complex.

The main contribution of this chapter is that we propose a methodology for designing contracts and control mechanisms for inter-organizational economic exchanges, in particular between enterprises in (virtual) network organizations.

This chapter is structured as follows. In Section 2 we introduce a methodology to design control mechanisms. The methodology consists of analysing *ideal* and *sub-ideal* situations in economic exchanges, and designing control mechanisms to prevent sub-ideal situations. Subsequently, we address the modelling of the ideal situation in Section 3, and propose some ideas for modelling sub-ideal situations and control mechanisms in Section 4.

[2] E.g., [LS03].
[3] E.g., [Luo01].

2 A Methodology for Designing Control Mechanisms

The main purpose of control mechanisms in electronic commerce is to ensure that all value exchanges between enterprises take place as planned or stated in a contractual agreement. There are many theories dealing with control mechanisms. The main elements of every control system[4] are the entity being controlled, detection of the violation, measurement of the severeness of the violation, and behaviour alteration. Similarly, control mechanisms can be seen as measures to prevent violation of obligations. Raskin, Tan, and van der Torre distinguish in [RTv96] two viewpoints on the behaviour of actors with regard to contractual obligations: either there is (1) no violation of an obligation by the actor, which is called *ideal* behaviour, or (2) there is a violation of an obligation, which is called *sub-ideal* behaviour. Thus, we suggest that a methodology for designing control mechanisms should include at least the following subsequent steps:

1. Design of a business value model (ideal situation)
2. Analysis of contractual obligations and their possible violations (sub-ideal situation)
3. Design of control mechanisms (prevention of sub-ideal situation)

A business value model shows *what* is actually being controlled or governed by a contract. This is an essential element of every control system.[5] In electronic commerce, such a value model is often unclear because most models are innovative. Consequently, it is necessary to understand the exchanges of economically valuable objects between actors before starting to design controls for these exchanges. We assume that a business value model only deals with ideal behaviour; that there is no violation of obligations. In practice, it is sufficiently difficult to design and understand such a model [GA03], and the addition of obligation issues would unnecessary complicate such a task. Therefore, we divide the design of a commercially sound value model from the analysis of possible violations and the design of control mechanisms.

After it is understood what value exchanges actually have to be controlled, possible violations are analysed and control mechanisms are designed. According to the description of a control system in Anthony and Govindarjan,[6] it is necessary to measure what is going on in the process being controlled, and to determine the significance of what is happening. In our terms, before introducing control mechanisms, we need to know *how the ideal situation can be violated*, i.e., the *sub-ideal situation* has to be modelled, and what the *severeness* of the violation is. In the next two sections we address the modelling of ideal and sub-ideal situation in electronic commerce.

[4] [AG03]
[5] [AG03]
[6] [AG03]

3 Modelling Business Value Models

A first step in designing an electronic commerce business case, is the development of a business value model, focusing on how value creation, distribution and consumption happens in a network of actors.[7] We use the e^3-*value* methodology for designing a business value model.[8]

The e^3-*value* methodology is a graphical tool for supporting the design of business models. Moreover, the tool enables sensitivity analysis of the profitability of such business models (for further details see [Gor02]). In particular, the e^3-*value* methodology provides modelling concepts for showing which parties exchange things of *economic* value with whom, and what is expected *what* in return. The methodology has been applied in a series of case studies to design value models of network organisations in media, banking, insurance and telecommunications.

Most of the currently available design methodologies for business models focus on process design rather than value design. They only focus on analysing the business processes that are needed to realise a value proposition. There are a few value chain design methodologies, which provide concepts for describing value constellations. For example, the AIAI Enterprise conceptual framework [GM99] or the Resource Event Agent (REA) [UKMZ98] conceptual framework. However, these frameworks are mainly descriptive theories, and do not support the value chain design process. Other business modelling methodologies[9] offer only generic conceptual frameworks, but lack a precisely defined ontology that is required for modelling the relevant details of a business model. (The ontology of e^3-*value* will be explained below.) Tapscott et al.[10] provide a graphical tool to represent economic exchanges between enterprises, however, compared to e^3-*value*, it has several drawbacks; e.g., it has no notion of economic activity, does not allow the profitability analysis of the economic exchanges, and is not based on a precisely defined ontology.

We briefly describe the core concepts of the e^3-*value* methodology. In Figure 1 a buyer obtains goods from a seller and offers money in return. According to law, the seller is obliged to pay the value-added tax (VAT). This is conceptualised by the following e^3-*value* constructs:
(Note: The legend upper part is only for explanatory purposes and is not part of the e^3-*value* modelling technique itself.)

Actor. An actor is perceived by its environment as an independent economic (and often legal) entity. An actor aims to make a profit or increase its utility. In a sound, sustainable business model each actor should be capable of making profit. The example shows a number of actors: a *buyer*, a *seller*, and a *tax administration*.

[7] See [PG03] for an overview.
[8] [Gor02,GA03]
[9] E.g., [AZ01], [OP02], and [PKT01].
[10] [TTL00]

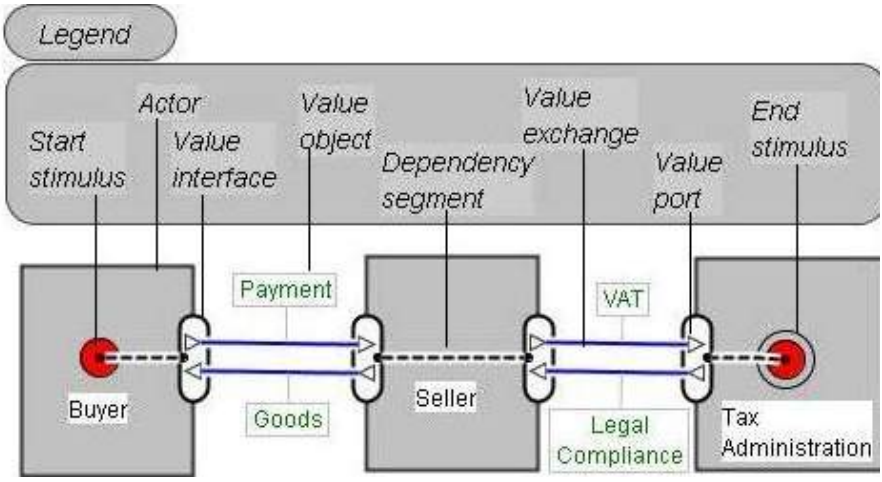


Fig. 1. e^3-*value* model of a purchase with tax payment

Value Object. Actors exchange value objects, which are services, products, money, or even consumer experiences. The important point here is that a value object is *of value* for one or more actors. *Good* and *payment* are examples of value objects, but *legal compliance* to pay tax is also a value object.

Value Port. An actor uses a value port to show to its environment that it wants to provide or request value objects. The concept of port enables to abstract away from the internal business processes, and to focus only on how external actors and other components of the business model can be 'plugged in'.

Value Interface. Actors have one or more value interfaces, grouping reciprocal, opposite-directed value ports. A value interface shows the value object an actor is willing to exchange, *in return for* another value object via its ports. The exchange of value objects is atomic at the level of the value interface.

Value Exchange. A value exchange is used to connect two value ports with each other. It represents one or more *potential* exchanges of value objects between value ports.

Using these concepts we can explain who wants to exchange values with whom, but we cannot yet explain what happens in response to a particular end-consumer need. For this purpose we include in the value model a representation of *dependency paths* (based on [Buh98]) between value interfaces. A dependency path connects the value interfaces in an actor and represents triggering relations between these interfaces. A dependency path consists of dependency nodes and segments.

Dependency Node. A dependency node is a stimulus (represented by a bullet), a value interface, an AND-fork or AND-join (short line), an OR-fork or OR-join (triangle), or an end node (bull's eye). A stimulus represents a consumer need, an end node represents a model boundary.

Dependency Segment. A dependency segment connects dependency nodes and value interfaces. It is represented by a link.

Dependency Path. A dependency path is a set of dependency nodes and segments that leads from one value interface to other value interfaces or end nodes of the same actor. A path indicates that if values are exchanged via a value interface, then other value interfaces connected by the path also have to exchange values.

A basic assumption of the e^3-*value* methodology is the *Principle of Reciprocity*, which means that if an actor offers something of value to another actor, this actor always gets something in return what s/he wants. In other words, in e^3-*value* models it is assumed that all actors behave correctly, and hence only ideal situations can be represented. The violation of the principle of reciprocity (e.g. an actor obtaining something without paying for it) can be seen as a violation of an obligation, which would lead to incorrect e^3-*value* models with a value interface with only one incoming or outgoing value object, i.e. delivering goods and not receiving a payment in return. In the next section we extend e^3-*value* with additional concepts and rules to enable the representation of such sub-ideal situations as violations of the principle of reciprocity.

4 Modelling Sub-Ideal Situations

A large number of theories about modelling obligations and their violations have been developed in the area of deontic logic (for an overview see [MW93].) In Raskin et al.[11] and in Tan & Thoen,[12] a method was introduced to model obligations using Petri Nets. A Petri net is a graphical formalism for modelling and analysing discrete dynamic systems. It was suggested to model procedures and processes within and between organisations with extended Petri Nets, which model preference relations for different executions of transactions between actors.

A method similar to the one described in [RTv96] can be used to extend e^3-*value* in such a way that it becomes possible to represent obligations and violations of obligations. We will call this extension e^3-*value+*. Violations of obligations in e^3-*value+* are modelled by so-called *sub-ideal* paths. These are typically paths for which the principle of reciprocity does not hold. For example, although in a contract it was agreed that the buyer would pay the seller in return for the delivery of the goods, there can be a sub-ideal path in which the seller does deliver the good, but the buyer does not pay within the

[11] [RTv96]
[12] [TT98a]

agreed period of time. So, the first extension we have to make to the standard e^3-*value* tool is that we omit the principle of reciprocity and allow for this type of sub-ideal paths. The second extension is that we assign weights to the segments of a path. As in [RTv96], these weights represent a kind of fines. So, if a segment has a non-zero positive weight it reflects that this is a kind of sub-ideal behaviour that is fined with a penalty. A zero weight means that the segment does not violate the terms of the contract.

Figure 1 shows an e^3-*value* model, representing the ideal situation. In Figure 2 we use an e^3-*value*+ to model sub-ideal situations that can happen when one of the actors violates the contract (e.g., does not respect economic reciprocity).

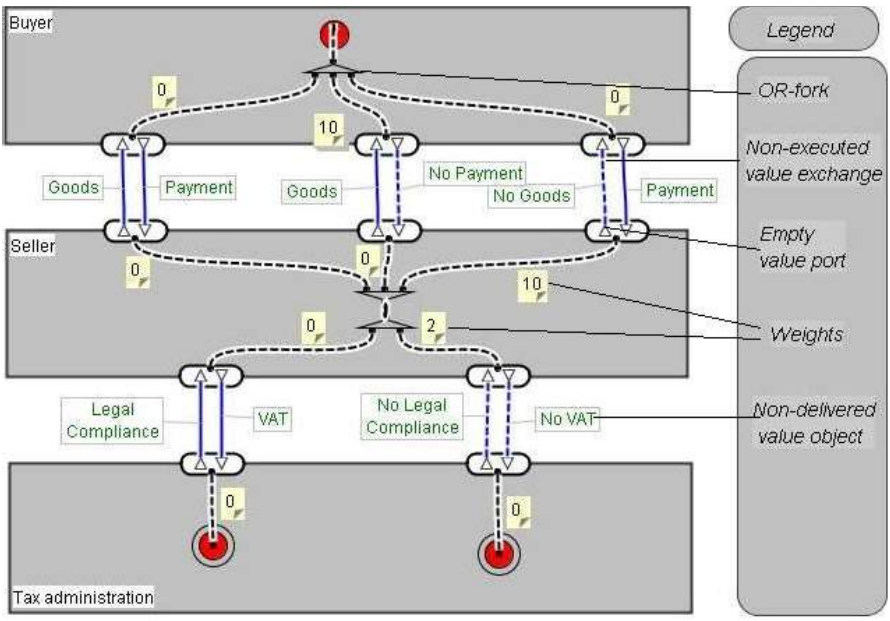


Fig. 2. Modelling violated obligations in e^3-*value*+

Modelling these sub-ideal paths is typical for contract drafting when the contract partners negotiate about the possible problems and contingencies that can occur during the execution of the contract and mutually agree on additional clauses to cover risks of non-performance of the other party. Typically, these extra contract clauses also include certain financial compensations, for example a reduction of the price if the goods are delivered too late. In research on contracting it has been observed that this reasoning about sub-ideal paths, which is called contingency planning, is one of the most important, as well as most labour intensive, stages of contract negotiation. For example, in [McC63] it was observed that contingency planning is the most

useful part of the contract, because it learns the contract partners what problems they could encounter, what reasonable financial compensations are to be paid when they do occur, and how to minimise the likelihood that these problems will occur. One of the main purposes of e^3-*value+* is to support this contingency planning process in contract negotiations.

Compared to Figure 1, in Figure 2 we now have two additional value exchanges between the buyer and the seller, which represent two possible sub-ideal situations: (1) the seller delivers goods, and the buyer does *not* pay, and (2) the seller does *not* deliver goods, while the buyer pays. The first is bad behaviour of the buyer, and the second is bad behaviour of the seller. As a result, instead of one scenario path in the ideal situation in Figure 1, we now have in Figure 2 three different alternatives, modelled as one ideal and two sub-ideal paths coming out of the OR-fork. In addition, we also assigned weights to each of the path segments. These weights reflect the severeness of the violation. For example, the buyer can face three situations: to pay and to receive goods, to receive goods and not pay, and to pay, but not to receive goods. The worst behaviour of the buyer is to receive goods, but not to pay, because then the buyer violates his/her obligation to pay. By assigning a higher weight to the segment, which indicates the worst case for the buyer, the incentive is modelled for the buyer not to take this sub-ideal path.

As for the exchanges between the seller and the tax administration, the ideal situation is modelled as the seller paying tax and the tax administration confirming a legal compliance. In the sub-ideal situation the principal of reciprocity can be only violated by not paying tax (we assume that a tax office cannot violate it by punishing the seller even if the tax is paid). Note that the violations of obligations in exchanges between the seller and the tax administration are independent of violations of obligations between the seller and the buyer, and the other way around. The model includes a number of possible sub-ideal paths: (a) the goods are delivered, the fee is not paid, and the tax is paid, (b) the goods are not delivered, the fee is paid, and the tax is paid, (c) the goods are delivered, the fee is not paid, and the tax is not paid, (d) the goods are not delivered, the fee is paid, and the tax is not paid, and (e) the goods are delivered, and the fee is paid, but the tax is not paid. Below we describe the extensions of e^3-*value+* in more detail.

4.1 Extension 1: The Violation of the Principle of Reciprocity

The ideal situation between the buyer and the seller is represented by value exchanges denoting transfers of goods and fees between them. We model a sub-ideal situation as a violation of the principle of economic reciprocity. The failure to deliver a value object is the violation of the principle of reciprocity. In e^3-*value* the principle of reciprocity is "hard-wired", and the violation of the principle of reciprocity in e^3-*value+* is reflected in the changes of the following e^3-*value* concepts:

Value Interface. A value interfaces consists of groups of in-going and out-going value ports. It shows the value object an actor is willing to exchange *in return for* another value object via its ports. The failure to deliver value object results in an incomplete set of value exchanges coming out and in the value interface.

Value Port. Value object is exchanged between actors via value ports of the value interface. If the principle of reciprocity is violated, no value object and no value exchange goes through the value port. We call this value port an *empty value port*.

Value Object. The violation the principle of reciprocity results in new types of value objects. Basically, there are two types of value objects: *delivered value object* and *non-delivered value object*.

Value Exchange. Value exchange is closely related to value interface, and the violation of the principal of reciprocity will result in a value exchange that it is not executed. Thus, there are also two types of value exchanges: *executed value exchanges* and *non-executed value exchanges*.

To graphically represent the violation of the principle of reciprocity we use different types of value exchanges in e^3*-value+*. Executed value exchanges are represented with solid lines, and non-executed ones with dashed lines. The value ports that are connected to non-executed value exchanges are empty. The names of non-delivered value objects are also changed depending on the corresponding delivered value objects (like, "Payment" for the delivered value object and "No Payment" for the non-delivered one).

4.2 Extension 2: Weights of Value Ports, Path Segments, and Sub-Paths

We modelled five violation situations, but they are not of the same severeness. For example, one can argue that even if the buyer did not pay the seller for the delivered goods, for the seller it is still better to pay tax rather than not to pay tax, because VAT is based on the seller's invoice, and not on whether the seller actually received the payment from the buyer.

The idea of differentiating between levels of severeness of sub-ideal situations is well known and widely applied in laws and policies. The systems of various punishments for different levels of crimes are an example.

We model the difference in severeness of violations by assigning different weights to value ports. Weights represent fines. The basic heuristics is the higher the weight, the more severe the violation. Weights are assigned to *outgoing value ports* (value ports, having *outgoing* value objects). It is presupposed that the failure to deliver the value object is a responsibility of the party that offers this object, not the party that accepts it. A non-empty value port has a zero weight, notifying that there is no violation of the terms of the contract, and, therefore, no fine is assigned. An empty value port has a non-zero positive weight, which reflects a kind of sub-ideal behaviour that is fined with a penalty. Weights are accumulated at the segment, connected

to the value interface. We define a weight of the segment as a function of the corresponding outgoing value ports.

The following definitions and rules characterize the modelling weight in e^3-*value+* methodology. The ordering of seriousness of violations for segments is done at the level of the actor. First, we introduce the following heuristics to distinguish the weights assigned in ideal and sub-ideal situations.

Property 1 (The weights of non-empty value ports are equal to zero). Let V_i, *i=(1,n),* be a non-empty outgoing value port, n is the number of non-empty outgoing value ports in the value model, w_i is the weight of the value port V_i. Then $\forall i \in (1,n)\ w_i = 0$.

Property 2 (the weights of empty and non-empty value ports). Let V_i, *i=(1,n),* be a non-empty outgoing value port, n is the number of a non-empty outgoing value ports in the value model, w_i is the weight assigned to the value port V_i; V_j, *j=(1,m),* is an empty outgoing value port, j is the number of such value ports in the value model, w_j is a weight function of the value port V_j. Then

$$\forall i \in (1,n),\ j \in (1,m), \quad w_i < w_j$$

Further, we define the weight of a segment as a function of the weights of the connected value interface.

Definition 1 (the weight of a segment). Let (I_i, S_i) be a tuple ranging over $i = (1,n)$, where n is the number of all the value interfaces in a specific value model, I_i is a value interface and S_i is a segment connected to this interface, $V_{ij} \in I_i$,with *i=(1,n), j=(1,m),* is an *outgoing* value port (a value port having *outgoing* value object), j is a number of *outgoing* value ports in the value interface I_i, w_{ij}, with $w_{ij} \geq 0$, is the weight of the value port V_{ij}. We call

$$w_i = \sum w_{ij}, \quad i = (1,\ldots,n), j = (1,\ldots,m)$$

the *weight of the segment* S_i.

This definition says that the weight of the value interface is the sum of the weights of value objects exchanged via this value interface. For example, in Figure 3, the buyer and the seller exchange goods, and the buyer has to make two payments: one for goods, and the other for delivery. The model on the left shows that the buyer receives two fines: the weight 10 is assigned to the outgoing value port for not paying goods, and the weight 2 is assigned to the outgoing value port for not paying for the delivery. Thus, the buyer accumulated 12 points of fine as a segment weight. The model on the right shows that the buyer did not pay the delivery, thus receiving 2 points of fine as in the previous case, but the value port with the "Goods payment" object receives has a zero weight, because the buyer pays for the goods; in total the segment weight of the buyer is 2.

In the other examples in this chapter the models do not contain such a value interface with more than one outgoing value port, and therefore, the segment weight equals the weight of the only outgoing value port.

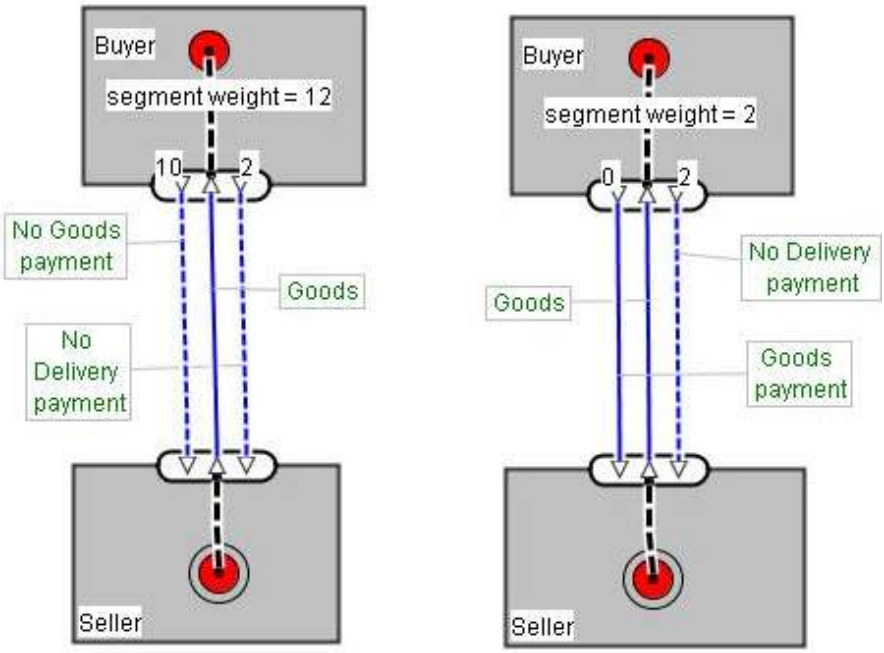


Fig. 3. Weight of the segment as a sum of the weights of value ports with outgoing value objects

From Properties 1 and 2, and Definition 1 it follows that the weight of a segment connected to a value interface with empty value ports is larger than the weight of the segment connected to a value interface with non-empty value ports, which reflects a kind of sub-ideal behaviour that is fined with a penalty.

A path can be assigned a total weight, which is the sum of the weights of the segments that the path consists of.

Definition 2 (the weight of a path). Let S_i, *i=(1,n)* be a collection of segments from a scenario path P, n is the number of segments in the scenario path P, w_i, with $w_i \geq 0$, is the weight of the segment S_i. We call $W = \sum w_i$, *i=(1,n)* the *weight of the path* P.

In the ideal situation the total weight of the path will be zero. In sub-ideal situations the total weight of the path will be a non-zero positive number; we call these paths *sub-ideal paths*. The weight of the path reflects the seriousness of violations in the path from global perspective. We use the concept of the path weight to distinguish between the ideal and sub-ideal paths.

Observation 1. The weight of the ideal path equals 0.

Observation 2. (weights of ideal and sub-ideal paths) The weight of the ideal path is less than the weight of any sub-ideal path. Let P_s be an ideal path with total weigh W_s, P_i, $i = (1, n)$ be a set of sub-ideal paths

with total weight W_i, $i = (1, n)$, i is a number of sub-ideal paths in the value model. Then $\forall i \in (1, n), \quad W_s < W_i$.

Assigning different weights to different sub-ideal situations is part of the modelling process, just as is the introduction of sub-ideal paths. The e^3-*value*+ methodology does not prescribe the contract partners, which sub-ideal paths are relevant to consider, or which specific numbers have to be assigned to a weight. These are modelling issues that contract partners have to agree upon among each other. However, e^3-*value*+ provides concepts and rules to structure this modelling process.

For example, in Figure 2, the weights of segments that are connected to value interfaces with non-empty outgoing value ports are zero. In the exchange between the buyer and the seller when the goods are delivered (value object "Goods") and the fee is not paid (value object "No Payment"), a non-zero positive weight (in this case 10), is assigned to the segment connected to the empty outgoing value port with the non-delivered value object "No Payment". Similarly, the segments leading to empty outgoing value ports with the associated non-delivered objects "No Goods", and "No VAT" have non-zero positive weights 10 and 2, respectively. In assigning weights in Figure 2 we used the heuristics, that (1) the same penalty for the seller of not delivering goods as for the buyer of not paying the goods, and (2) for the seller, not delivering goods is less serious than not paying taxes.

To keep things simple we use absolute numbers here to explain the method. Of course, these weights are rather arbitrary. In typical contract negotiations the contract partners agree among each other how to assign these weights (and typically translate these weights in financial compensations). For example, the seller might require that in case of late payment the buyer should pay a certain percentage extra on top of the sales price. A more sophisticated representation could be to use a partial ordering of all the weights rather than absolute numbers. This partial ordering will be investigated in future research.

Having weights assigned to segments it is possible to distinguish the severeness of violation from the viewpoint of every single actor in the model. Table 2 represents five sub-ideal paths of Figure 2. The last three columns include the actors. Each actor in each sub-ideal situation accumulates weights, which are assigned to the individual segments.

For example, the "No Payment" value object is assigned to the buyer, because the value interface with outgoing value object "No Payment" belongs to the buyer, notifying that the buyer did not pay (see Figure 2). Consequently, we can say that for the buyer sub-ideal situations (a) and (c) are the worst because they have the highest weights 10, while other paths are equal to the path of the ideal situation. Similarly, for the seller, the worst sub-ideal situation is (d): it has the highest total weight 12 (no goods are delivered and no tax is paid). The situation (c), when the buyer did not pay for the goods, and the seller did not pay taxes, is worse for the buyer than for the

Table 1. Actor's view on modelling severeness of violation weights for different sub-ideal situations

Sub-ideal path	Non-delivered value objects		
	Buyer	Seller	Tax administration
a) The goods are delivered, the fee is not paid, and the tax is paid	No Payment w = 10	w = 0	w = 0
b) The goods are not delivered, the fee is paid, and the tax is paid	w = 0	No Goods (w = 10)	w = 0
c) The goods are delivered, the fee is not paid, and the tax is not paid	No Payment w = 10	No VAT (w = 2)	w = 0
d) The goods are not delivered, the fee is paid, and the tax is not paid	w = 0	No Goods (w = 10) No VAT (w =2)	w = 0
e) The goods are delivered, and the fee is paid, but the tax is not paid	w = 0	No VAT (w = 2)	w = 0

seller: the buyer accumulates the weight 10, while the seller has only 2. For the tax administration every modelled situation is equal to ideal: the tax administration is modelled as a governing institution that cannot be punished for violation of obligations.

5 Distinguishing Different Types of Control Mechanisms

Management control systems are applied by the management of an organization to achieve strategic objectives. Management control systems include many types of control mechanisms. An important type of control mechanisms are so-called Formal Control Mechanisms, which can be subdivided into Outcome Control mechanisms and Behaviour Control mechanisms (see [DT98] and [Dek03]). Outcome control mechanisms are characterized by the definition of goals for task performance, in particular incentive and reward systems to encourage desired behaviour or fine systems to discourage undesired behaviour. Behaviour control mechanisms are characterized according to Das and Teng by "reporting and checking devices, written notice of any departure from the agreement, accounting examination". Hence the use of trade documents to monitor the actual behaviour of buyers and sellers is a typical example of a behavioural control mechanism.

In the e^3-*value*+ methodology we can clearly define the outcome control mechanisms. The weights can be considered as a fine system to discourage

undesirable sub-ideal behaviour. What is still lacking is behaviour control mechanisms to monitor the actual performance of the behaviour. Here one could think about the use of a bank statement as evidence by the seller that a payment was made. In the current version of e^3-*value+* methodology these evidentiary documents are not modelled. In particular the exchange of these documents. This would require the model of business processes between and within organisations.

The efficiency of control mechanism is related to the cost of the implementation of control mechanism. If an actor considers implementing a control mechanism, the cost of it should be not higher than the potential losses of the sub-ideal behaviour.

6 Conclusions

We have introduced the e^3-*value+* design methodology, which is an extension of the e^3-*value* methodology for designing business models for virtual network organisations. The e^3-*value+* design methodology is being developed, in particular, for modelling inter-organisational control mechanisms. The e^3-*value+* methodology supports the analysis of violations of obligations as well as the corresponding design of control mechanisms to minimise the likelihood of these violations. The fine systems, described in this chapter, can be considered as an incentive for actors not to misbehave, and hence as a type of control mechanism. This methodology supports the contingency planning phase of contract drafting, because it helps the contract partners to understand what violations of contractual obligations they could encounter, what reasonable financial compensations are to be paid when they do occur, and how to minimise the likelihood that these problems will occur.

References

[AG03] R. Anthony and V. Govindarjan, *Management control systems*, 11th ed., McGraw-Hill/Irwin, New York, NY, 2003.

[AZ01] R. Alt and H. D. Zimmerman, *Preface: Introduction to special section — business models*, Electronic Markets **11** (2001), no. 1, 3–9.

[Bar01] D. Barkai, *Technologies for sharing and collaborating on the net*, Proceedings of the 1st International Conference on Peer-to-Peer Computing (Lingkoping, Sweden), 2001.

[Buh98] R. A. J. Buhr, *Use case maps as architectural entities for complex systems*, IEEE Transactions on Software Engineering **24** (1998), no. 12, 1131–1155.

[Dek03] H. Dekker, *Control of inter-organizational relationships: Evidence on appropriation concerns and coordination requirements*, Accounting, Organizations and Society **29** (2003), 27–49.

[DS97] Aspassia Daskalopulu and Marek Sergot, *The representation of legal contracts*, AI and Society **11** (1997), no. 1, 6–17.

[DT98] T. K. Das and B. S. Teng, *Between trust and control: Developing confidence in partner cooperation in alliances*, Academy of Management Review **23** (1998), no. 3, 491–512.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke, *The anatomy of the grid: Enabling scalable virtual organizations*, International Journal of Supercomputer Applications and High Performance Computing **15** (2001), no. 3.

[GA03] J. Gordijn and J. M. Akkermans, *Value based requirements engineering: Exploring innovative e-commerce ideas*, Requirements Engineering Journal **8** (2003), no. 2, 114–134.

[GM99] Guido L. Geerts and William E. McCarthy, *An accounting object infrastructure for knowledge-based enterprise models*, IEEE Intelligent Systems **14** (1999), no. 4, 89–94.

[Gor02] J. Gordijn, *Value-based requirements engineering: Exploring innovative e-commerce ideas*, Ph.D. thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2002, `http://www.cs.vu.nl/~gordijn`.

[LS03] H. Ludwig and M. Stolze, *Simple Obligation and Right Model (SORM) for the runtime management of electronic service contracts*, Proceedings of the CAISE–WES, 2003.

[Luo01] Y. Luo, *Contract, cooperation, and performance in international joint ventures*, Strategic Management Journal **23** (2001), 903–919.

[McC63] S. McCauley, *Non-contractual relations in business: A preliminary study*, American Sociological Review **28** (1963), 55–67.

[MCC98] D. H. McKnight, L. L. Cummings, and N. L. Chervany, *Initial trust formation in new organizational relationships*, Academy of Management Review **20** (1998), no. 3, 472–490.

[MW93] J.-J.Ch. Meyer and R. Wieringa (eds.), *Deontic logic in computer science*, Wiley, Chichester, UK, 1993.

[OP02] A. Osterwalder and Y. Pigneur, *An e-business model ontology for modeling e-business*, Proceedings of the 15th Bled Electronic Commerce Conference (Kranj, Moderna Organizaja), 2002, pp. 1–12.

[Ouc79] W. G. Ouchi, *A conceptual framework for the design of organizational control mechanisms*, Management Science **25** (1979), 833–848.

[PG03] A. G. Pateli and G. M. Giaglis, *A framework for understanding and analysing e-business models*, Proceedings of the 16th Bled Electronic Commerce Conference: eTransformation (Kranj, Moderna Organizaja) (R.T. Wigand, Y.H. Tan, J. Gricar, T. Pucihar, and T. Lunar, eds.), 2003, pp. 329–348.

[PKT01] O. Petrovic, C. Kittl, and R. D. Teksten, *Developing business models for e-business*, Proceedings of the 2nd International Conference on Electronic Commerce, 2001.

[RTv96] J. Raskin, Yao-Hua Tan, and L. van der Torre, *How to model normative behavior in Petri Nets*, Proceedings of the 2nd Modelage Workshop on Formal Models of Agents (Sesimbra, Portugal), Modelage'96, 1996, pp. 223–241.

[TT98a] Yao-Hua Tan and Walter Thoen, *A logical model of transfer obligations in trade contracts*, Accounting, Management and Information Technologies **8** (1998), no. 1, 23–38.

[TT98b] ———, *Towards a generic model of trust for electronic commerce*, International Journal of Electronic Commerce **3** (1998), no. 1, 65–81.

[TT01] ———, *A survey of electronic contracting related developments*, Proceedings of the 14th Bled Electronic Commerce Conference (Kranj, Moderna Organizaja), 2001.

[TTL00] D. Tapscott, D. Ticoll, and A. Lowy, *Digital capital: Harnessing the power of business webs*, Nicholas Brealy Publishing, London, UK, 2000.

[UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios, *The enterprise ontology*, The Knowledge Engineering Review **13** (1998), no. 1, 31–89.

Sim-I-Space: An Agent-Based Modelling Approach to Knowledge Management Processes

Max Boisot[1], Ian MacMillan[2], Kyeong Seok Han[3], Casey Tan[4], and Si Hyung Eun[5]

[1] Universitat Oberta de Catalunya, Barcelona, Spain,
boisot@attglobal.net
[2] The Sol Snider Entrepreneurial Research Center, The Wharton School, University of Pennsylvania, Philadelphia, PA, USA,
macmillan@wharton.upenn.edu,
[3] Soongsil University, Seoul, 156–743, Korea,
kshan@ssu.ac.kr
[4] Singapore Airlines, Singapore,
case_tan@hotmail.com
[5] The Sol Snider Entrepreneurial Research Center, The Wharton School, University of Pennsylvania, Philadelphia, PA, USA,
silver92@empal.com

Abstract. In the chapter we offer a verbal description of Sim-I-Space, an agent-based model that operationalises key features of a conceptual framework: the Information-Space (I-Space). The I-Space relates the speed and extent of information flows between agents to how far their messages have been structured through acts of codification and abstraction. The more structured a message, the faster and more extensively it diffuses to other agents—intentionally or not.

Following a brief introduction, the paper divides into two sections. Section 2 describes the models architecture, the agents, the nature of the knowledge assets that they create, articulate, and trade in, and the types of the interactions—trading, licensing, joint-venturing, merging and acquiring—that agents can engage in. Section 3 presents the main components of Sim-I-Space, namely, agent characteristics, agent knowledge, and agent interaction. Two appendices—A and B—describe the model variables and provide a more detailed model specification.

1 Introduction

In this paper we offer a verbal description of the Sim-I-Space simulation model. The model is designed to operationalise some of the main features of the Information Space or I-Space [Boi98].

The paper is structured into two main parts as follows. In the first part we look at the overall architecture of Sim-I-Space that brings together three components:

- agents,
- knowledge assets, and
- agent interactions.

In the second part we examine in more detail each of the model components. There are two appendices. In Appendix A we describe the variables used in the model and identify the input parameters used in the model. In Appendix B we provide a more detailed specification of the model, with illustrative examples.

2 Model Architecture

Sim-I-Space is a multi-agent simulation characterized by mixture of competition and collaboration. Although built in part on a Swarm platform, in the limited number of time periods it runs, and given that agents make their decisions at random—i.e., they do not learn—it only exhibits limited elements of evolutionary behavior. Survival is the aim of individual agents in the simulation. The rents that agents earn provide them with the means to survive. If agents run out of money they are 'cropped'. They can quit while they are still ahead. The overall value of a given simulation run is the sum of the rents earned by all agents during the run. The social welfare generated by the simulation is the sum total of all knowledge created in the course of the simulation and then diffused out to 'society'. Note that 'society' is located outside the simulation. The price paid by 'society' for this social welfare is the cumulative rent earned by the agents inside the simulation, i.e., the price paid by 'society' turns out to be the value of the simulation.

How does Sim-I-Space implement and embody the concepts of the I-Space? The I-Space is a conceptual framework for analyzing the nature of information flows between agents as a function of how far such flows have been structured through processes of codification and abstraction. Such flows, over time, give rise to the creation and exchange of knowledge assets. Where given types of exchange are recurrent, they will form transactional patterns that can be institutionalized. In Sim-I-Space, we focus on the creation and exchange of knowledge assets alone without concerning ourselves with the phenomenon of recurrence and institutionalization. In later versions of the model, recurrence will become our central concern.

Sim-I-Space is populated with agents that carry knowledge assets in their heads. Each of these knowledge assets has a location in the I-Space that changes over time as a function of diffusion and obsolescence processes as well as of what agents decide to do with them. These have the possibility of exchanging their knowledge assets in whole or in part with other agents through different types of dealing arrangements.

Natural selection is at work in Sim-I-Space at two levels. At one level, agents survive by learning to make good use of their knowledge assets. They can make use of these assets directly to earn rents, or they can make indirect

use of these assets by entering into trades with other agents who will then use them directly. Agents that fail to make good direct or indirect use of such assets in a timely fashion fail to earn the minimum rent required to survive and are selected out of the simulation, i.e., they are 'cropped'. At another level, knowledge assets, in turn, and somewhat like Dawkins's memes [Daw82,Daw89], survive by inhabiting the heads of many agents. If they fail to occupy at least one agent's head, they die out and the knowledge associated with the asset disappears from the simulation as a resource.

Existing agents have the option of quitting the simulation while they are ahead and before they are cropped. Conversely, new agents can be drawn into the simulation if the environment becomes sufficiently rich in opportunities for earning rents.

The rate of entry and exit of new agents into the simulation are based on the difference in mean rents between between two periods. The rate of entry and exit is a parameter that is set at the beginning of the simulation for every % (percentage) change in mean rents. Change in the entry and exit rates is a function of % change in mean rents. In this way one can control the level of market turbulence—of creative destruction, if you will—that is generated by the performance of existing players.

We start by discussing the agents and then turn to a discussion of the nature of their knowledge assets. This is followed by a brief discussion of agent interactions.

2.1 Agents

Sim-I-Space operates through a number of agents that, taken together, make up the diffusion dimension of the I-Space. In the model as developed, agents are intended to represent organizations—firms or other types of information-driven organizations—within an industrial sector. It would be quite feasible, however, with suitable parameter settings, to have the agents represent individual employees within a single firm and hence to simulate the behavior of individual organizations. It would also be possible to have an individual agent represent the behavior of a strategic business unit within a single firm. Conversely, one could run Sim-I-Space above the firm level and simulate knowledge flows within a population of industries.

As we have already seen, agents can enter or exit Sim-I-Space according to circumstances and can also be cropped from the simulation if their performance falls below a certain threshold. Agent entry and exit is an important source of variation within the simulation. Clearly, the population that is located along the diffusion dimension of the I-Space will vary in size at different moments in the simulation.

Agents aim to survive within the simulation and to maximize their wealth over the periods of the simulation. Wealth here is taken to be the sum of rent streams and of rent-generating knowledge assets. The first accumulate in a financial fund that is used to cover the expenses incurred in meeting

and transacting with other agents. The second accumulates in an experience fund that is used to finance the creation of new knowledge assets by moving existing ones in the I-Space. In sum, agents modify their wealth either by trading in knowledge assets they possess with other agents thereby enlarging or shrinking their asset base, or by creating new knowledge assets. They do this by moving around the I-Space in a learning process and by adding and then linking new knowledge assets to their existing stock [Boi98]. In this way they enhance their rent-generating potential. The details of how this is done are given under the heading of 'agent interaction'.

From the financial and experience funds, agents draw meetings and knowledge-investment budgets. Money that is not spent in a given period gets put back into the relevant fund where it accumulates. An agent's financial funds correspond to its tangible assets whereas its experience funds correspond to its non-fungible intangible assets. Each fund, or both, can be switched off with a toggle. The program can thus be made to behave in a modular fashion with agents surviving either through trading and collaboration with other agents alone, or through knowledge creation alone. An agent's preference for using one type of fund or for another—i.e., for trading in existing knowledge or for investing in knowledge creation—is set at the beginning of the simulation for all agents.

2.2 Knowledge Assets

In Sim-I-Space, knowledge assets are represented in network form. A knowledge network consists of a collection of elements and of relations between elements. We shall refer to the elements of the network as *nodes* and to the relations between elements as *links.* Nodes and links can be combined with certain probabilities[1] called *linkage probabilities.* A knowledge asset, then, can either be a node, a link between two nodes, or a set of interlinked nodes that can vary in size and complexity. Each node and each link varies in how far it has been codified, made abstract, or has been diffused to other agents. Thus each node and link has a unique location in the I-Space that determines its value to the agent and hence it's rents-generating potential. The more codified and abstract a knowledge asset in the I-Space, the greater its utility and hence the greater its value. Also, the less diffused a knowledge asset in the I-Space, the scarcer it is and hence, again, the greater its value. Different locations in the I-Space thus offer different rental potential to agents. These can be calibrated to reflect a variety of industry conditions—rates of knowledge obsolescence, tendencies to spillovers, etc. Agents can enhance the value of their knowledge assets—and hence their rents-generating potential—in two ways:

[1] Since in Sim-I-Space we do not specify the contents of nodes or links, we do not face the problem of establishing the *coherence* of networks created in this way. Future developments of the model will address this issue.

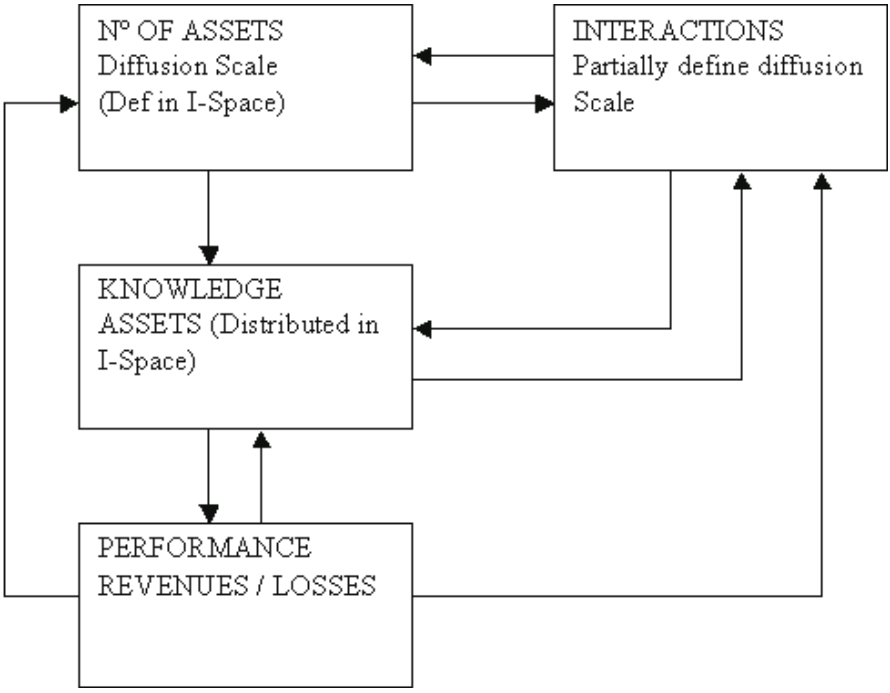


Fig. 1. Sim I-Space

- By investments in the Social Learning Cycle (SLC) that offer the possibility of changing the location of knowledge assets in the I-Space;
- By combining nodes and links into networks that can be *nested* and in this way building up more complex knowledge assets.

2.3 Agent Interactions

Agents meet each other throughout the simulation. The frequency of encounters can be varied. They can ignore each other or they can attempt to engage in different types of transactions. In the second case, they need to be able to inspect each other's knowledge assets in order to establish whether a transaction is worth pursuing. Having established that it is, they can either:

1. engage in straight buying as selling of knowledge assets;
2. license other agents to use their knowledge assets;
3. enter into a joint-venture with another agent by creating a new agent that is jointly owned;
4. acquire another agent and convert it into a wholly-owned subsidiary; or
5. merge with another agent, thus reducing the number of agents in the simulation.

The way that the simulation model maps onto the I-Space architecture is indicated by the flowchart of Figure 1. Having described the general architecture of Sim-I-Space, we now examine each of its components in more detail.

3 Model Components

In this section, we briefly describe the components of Sim-I-Space's architecture and show how I-Space elements and relationships are expressed in the model. We shall use the same basic headings to describe the components as we did to describe the model's overall architecture, namely, agent characteristics, agent knowledge, and agent interactions.

3.1 The Components of Agent Characteristics

Agent ID: a unique ID that is allocated sequentially identifies Agents in the model. No two Agents will have the same ID: Agent 0 is the first Agent created, Agent 1 is the second, and so on.

The Financial Fund and the Experience Fund: Agents possess resources that are used to cover their operating expenses. These resources are of two kinds: financial and experiential. Agents manage their resources by allocating rents to either a Financial or to an Experience fund, and by setting Financial and Experience Budgets. Speaking loosely, financial funds correspond to tangible assets such as cash in the real world whereas experience funds correspond to intangible assets such as know how. In effect, in the model, financial funds covers the costs of overt behaviors involving interactions between agents, whereas experience funds covers the costs of implicit agent behaviors associated with activities such as thinking and learning. Although in the real world overt and implicit behaviors are intimately intertwined, we can associate the first type of behavior with *operations* and the second with *development.* Specifically:

- *Financial Funds and Financial Budget*: Financial Funds correspond to tangible resources, such as cash, that are used to meet operating expenses, including the cost of meeting other Agents, trading for Assets, and paying out dividends. The Financial Budget is set by the Agent for each period to limit the amount of the Financial Funds that the Agent intends to expend each period.
- *Experience Funds and Experience Budget*: Experience Funds correspond to intangible resources, such as the experience gained from learning-by-doing, that are used to manipulate Assets – to increase/decrease their degree of Abstraction or Codification, or to combine Assets to create new Assets. The Experience Budget is set by the Agent to limit the amount of the Experience Funds that the Agent intends to expend each period.

Although Agents can survive without Experience Funds, they must not run out of financial funds otherwise they get cropped.

Active, Passive and Trading Sets: Agents have a finite memory and because of this, storing knowledge carries a cost. Agents manage the scarce resource that is their memory either by holding knowledge assets in one of two sets, or by discarding them altogether. The two sets are:

- *The active set*: an Agent's active set contains all the knowledge assets that are being actively utilized by the agent and that are generating rents in the period.
- *The passive set*: an agent's passive set contains all the knowledge assets held by an Agent that are not being actively utilized by the agent or generating rents in the period. Maintaining knowledge assets in the passive set reduces the cost of carrying them yet keeps them available for direct use or for trading purposes in later periods – i.e., they are held for their option value.

When agents meet, not all of their knowledge assets will necessarily be available for trading. When two agents meet for the first time, for example, the transaction costs associated with inspecting each other's knowledge assets – presentation, examination, evaluation, etc - will limit the process to the most abstract and codified of these assets. This much we established in our earlier discussion of the I-Space in chapter 4. Over time and with recurrent meetings, the degree of familiarity between any two Agents may increase and as a result the cost of inspecting each other's knowledge assets will decrease. The agents will then be able to inspect each other's more concrete and less codified knowledge assets, thus increasing the number of assets available for trading.

- *The Trading Set*: a trading set is a set of knowledge assets that an agent makes available for transactions with other agents. An agent will have more than one trading set and will present different trading sets to the different agents it meets according to the degree of prior familiarity with them.

Clearly, the more one agent is familiar with another, the less codified and abstract will be the assets that it presents for inspection and hence the larger the trading set it will present. Also, and by implication, the longer the time that will be allocated to the interaction.

Agent Memory: Agent memory stores the frequency of the historical encounters with other Agents. Based on this memory, an agent "decides" whether or not to interact with a given other agent, reflected in the probability of interaction between the two agents. Based on its previous memory the preference level of an agent to make a decision for the current meeting period will be

randomly assigned, ranging from a minimum preference level to a maximum preference level that can be preset to reflect the industry characteristics being simulated. In the meeting the agent will select one of several alternative actions: to trade, to license, to joint-venture, to merge or to walk away.

3.2 The Components of Agent Knowledge

Asset ID and Type: knowledge assets ("Assets") are either *nodes* or *links*. A unique ID number that is allocated sequentially identifies each Asset, i.e. no two Assets (nodes or links) will have the same ID. The first Asset will have ID 0, the second Asset will have ID 1, and so on.

Abstraction and Codification: Knowledge assets vary in how far they are codified and abstracted. In SimIspace, each Asset is given a discrete value for Abstraction and Codification that begins at 0 (respectively representing concrete and uncodified Assets), and goes up to 4. Users have to decide for themselves what meaning to give to these different levels of codification and abstraction. Users decide for themselves what meaning to give to these different levels of codification and abstraction.

Diffusion: Diffusion measures the number of agents in Sim-I-Space who have access to a given knowledge asset. Reflecting different degrees of codification and abstraction, each knowledge asset – in line with what is claimed by the I-Space - varies in the extent to which it is diffused among the agents in the model. Diffusion ranges from 0 to 3. Users have to decide for themselves what percentage of a given Agent population corresponds to these different levels of diffusion. It must be borne in mind that the size of the Agent population will keep on varying throughout the simulation.

Complexity: We saw earlier that knowledge could be represented as a network of interlinked nodes. Such networks vary in complexity depending on the number of nodes and the density of links between them. When parts of the network are tightly integrated with each other, they will themselves be represented by a node (see Figure 2). Given the network that 'nests' in such a node, it becomes a complex entity. This complex node can in turn participate in a higher-level network that is also capable of getting nested. We can thus represent the complexity of any network by counting the degree of nesting that takes place within its nodes. Such complexity begins at 0—i.e., no nesting takes place within the Assets - and has no upper limit so, a potentially infinite amount of nesting is possible. In the simulation, Assets can only combine with Assets of equal levels of complexity, and if nesting takes place at that level, the resulting Asset is of one larger complexity. Users have to decide for themselves what meaning to give to these different levels of complexity.

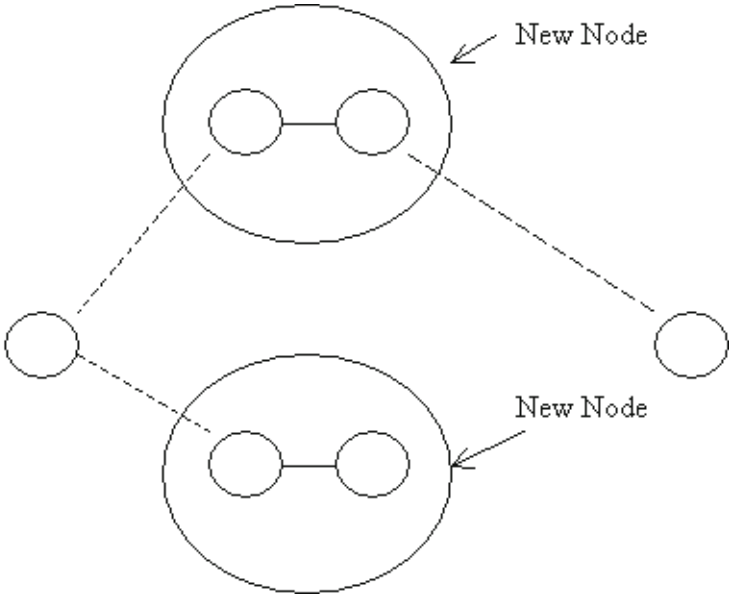


Fig. 2. Nested nodes

Base Rental Potential: Knowledge assets earn rents that vary with their position in the I-Space. The more closely they are located to the region of maximum value in the space—where codification and abstraction are at a maximum and where diffusion is at a minimum—the higher the rents that they can earn. The Base Rental Potential reflects the fundamental relationship between rents and the three attributes of abstraction, codification and diffusion possessed by a given knowledge asset. Thus, each I-Space location in the I-Space is associated with a Base Rental Potential that is applied to all Assets residing in that location.

Rental Multiplier: Actual rental levels will be a function of what, specifically, is being simulated in Sim-I-Space. Bread baking, for example, does not generate the same volumes of rents as biotechnology; no matter how asymmetrically distributed the knowledge assets might be. The Base Rental Potential is thus subject to a rental multiplier that determines the difference between the Rental Potential of two Assets of equal degrees of Abstraction, Codification, and Diffusion but used in two different industries. The Rental Multiplier is the link between the Base Rental Potential - a function of where a given Asset is located in the I-Space - and the actual rents earned per unit time in a given type of simulation. The Rental Multiplier takes into account a variety of factors such as complexity (increasing dramatically as Complexity

increases[2]), obsolescence (decreasing over time subject to the Obsolescence decay function described below), and the "jackpot effects" (extraordinary random increases in rents resulting from particular asset combinations). The rental multiplier applies to individual assets. It is a summary measure of how specific characteristics affect that asset's rents.

The effect of diffusion on the rental value of the Asset is non-linear – the loss in value from one additional Agent coming into possession of the Asset varies greatly depending on whether there was originally only one agent who owned the Asset, or if there were 100 agents who owned the Asset. In the former case, there is a sharp loss in value, and in the latter case, there will be only a marginal loss in value. To reconcile this difference there are two relevant values of Diffusion: 1) Nominal Diffusion (a continuous variable that measures the actual number of Agents that own the Asset that has no upper bound), and 2) Model Diffusion, a discrete variable derived from Nominal Diffusion and that is used in the I-Space Model to calculate the Rents Multiplier of the Asset. Model diffusion has an upper bound that is specified in the model to give varying degrees of granularity. For example, in a situation where Model Diffusion takes on the values of 0∼3, it might be determined as shown in Table 1.

Table 1. Nominal diffusion & model diffusion

Model Diffusion	**Nominal Diffusion (# of Agents that own the Asset)**
0	$1 \sim 2$
1	$3 \sim 8$
2	$9 \sim 26$
3	$27 +$

In the above example, it makes no difference in the Rents Multiplier whether 1 or 2 Agents own the Asset, but when a 3^{rd} Agent owns the Asset, the Rents Multiplier suddenly decreases and then remains steady until the Asset diffuses to the 9^{th} Agent, etc.

Carrying Costs: Maintaining Assets in a usable form imposes carrying costs on Agents. These are an exponential function both of the Assets' level of Complexity,[3] as well as of their degree of codification and abstraction. The

[2] By combining Assets, Agents create new Assets of higher Complexity (and Rents Multiplier). By doing so, Agents can reduce Carrying Costs, and usage of Agent memory (by replacing original Assets with the new combined Asset), while retaining some of the value of the original Assets as rental generators.

[3] Increasing Complexity thus has two conflicting effects—increasing both the Rental Multiplier, and the Carrying Cost.

higher a given knowledge asset's degree of codification and abstraction, the lower the level of entropy associated with it and hence the lower its carrying cost. However, we do not need to represent this second component of carrying costs as it is already reflected in the base rents potential. The main function of carrying costs in the simulation is to affect the choice of what assets will be placed in the active and passive trading sets. Carrying costs will be higher in the active set than in the passive set.

Creating New Assets: New Assets are created either through *moves* in the I-Space—i.e., through increases in codification, abstraction, impacting or absorption—or through the probabilistic *combination* of existing nodes and links (see discussion of the Linkage Probability Matrix in Appendices A and B). In the first case, new knowledge assets are created through a process of differentiation; in the second, through a process of integration. In either case, the "parents" of a new Asset are the nodes and/or links from which this Asset was derived. The Assets that agents are initially endowed with at the beginning of a Smart Asset run have no "parents".

The processes of codification, abstraction, impacting and absorp-tion—out of the six steps in the Social Learning Cycle—are those through which new knowledge assets are created in the I-Space. These are processes of *differentiation*. As the links between assets so created themselves increase in codification, so linked nodes can be nested and then collapsed into single nodes. These are processes of *integration*. We first discuss differentiation and then integration:

Differentiation. *Codification*: Codification of an Asset creates a new Asset of the same type and Complexity with the next higher Codification value. The new Asset inherits the Linkage Probabilities of its predecessor, with a random increase in all the existing non-zero values in the corresponding row (for Links) or column (for Nodes) in the Linkage Probability matrix. In this way, codification increases the linkage probabilities of a given knowledge asset's *existing* links

Abstraction: Abstraction of an Asset creates a new Asset of the same type and Complexity with the next higher Abstraction value. The new Asset inherits the Linkage Probabilities of its predecessor, but with an increase in the number of non-zero values randomly distributed in the corresponding row (for Links), or column (for Nodes) in the Linkage Probability matrix. In this way, abstraction extends a given knowledge asset's linkage probabilities to *new* links.

Absorption: Absorption of an Asset creates a new Asset of the same type and Complexity with the next lower Codification value. The new Asset inherits the Linkage Probabilities of its predecessor, with a random decrease in the all the existing non-zero values in the corresponding row (for Links) or col-

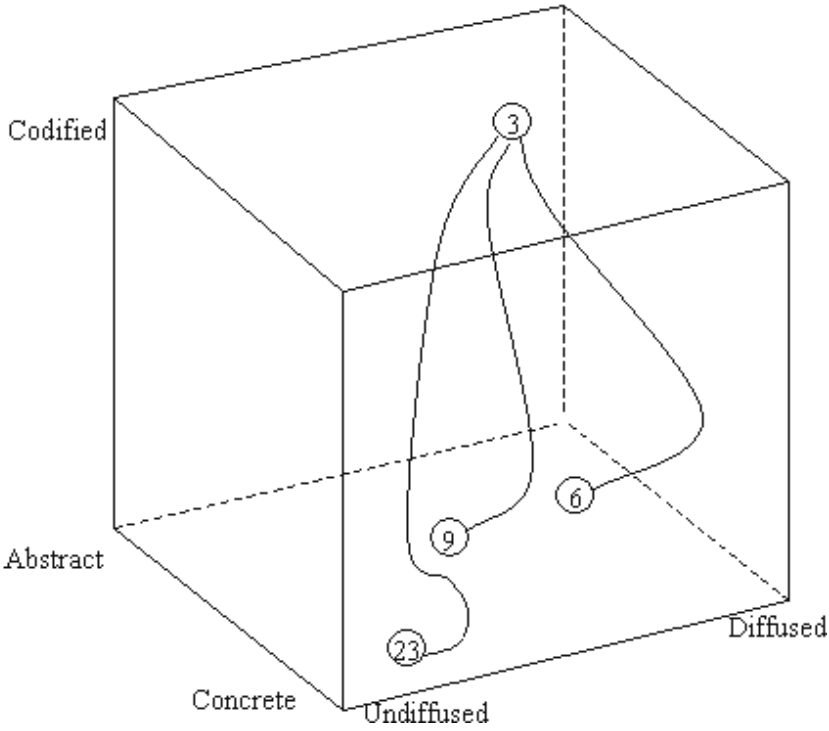


Fig. 3. Differential absorption and impacting in the I-Space

umn (for Nodes) in the Linkage Probability matrix. In this way, absorption *decreases* the linkage probabilities of a given knowledge asset's existing links

Impacting: Impacting of an Asset a new Asset of the same type and Complexity with the next lower Abstraction value. The new Asset inherits the original Linkage Probabilities of its predecessor, but with a decrease in the number of non-zero values randomly distributed in the corresponding row (for Links), or column (for Nodes) in the Linkage Probability matrix. In this way, abstraction *reduces* a given knowledge asset's linkage probabilities to new links.

There are several ways in which a given Asset can be codified, abstracted, absorbed or impacted, so that each Asset maintains memory of what are the possible results of the abstracting, codifying, impacting or absorbing process. Furthermore, the processes of abstraction, codification, impacting and absorption vary in their idiosyncrasy so that some results will be more widely diffused than others. In Figure 3 for example, it may be that the absorption of Node 3—i.e., its embedding and interpretation—will yield Node 6, Node 9 as well as Node 23. But, as indicated in the figure, 50% of the Agents

who successfully absorb Node 3 will create Node 6, 33% will create Node 9, and the remaining 17% will create Node 23. Clearly what we are dealing with here are three quite distinct interpretative schemas some of which are more idiosyncratic than others. The same argument applies to impacting, codification and abstraction

Integration. Integration relates one asset to another via links. The best way to visualize this is via a matrix representation of the I-Space.

The I-Space Matrix and I-Space Locations: The I-Space Matrix is a three-dimensional matrix representation of the I-Space whose axes stand respectively for abstraction, codification and diffusion. The I-Space Matrix is made up of discrete I-Space locations or 'cells' in which Assets reside. Each I-Space location is uniquely described by discrete values of abstraction, codification and diffusion, and can contain any number of Assets. Each I-Space Location in the I-Space Matrix is also associated with a Base Rental Potential (see above).

The Linkage Probability Matrix: New knowledge can be created in Sim-I-Space either by creating new links and nodes – this can be done by investing in the abstraction, codification, impacting and absorption of existing nodes and links - or by combining existing links and nodes in new configurations. In the latter case, nodes and links are brought together to create new knowledge assets by increasing the probabilities of linking them together. The linkage probability reflects the uncertainties associated with new knowledge creation. The linkage probabilities between a given node and a given link is indicated by their intersection on a Linkage Probability Matrix which lists all links on one side of the matrix and all nodes on the other. When an agent decides to increase its investment in new knowledge creation, this has the effect of either increasing the probability of creating individual nodes or links, or of increasing the linkage probabilities associated with a given link-node cell. Thus investing in a given knowledge asset does not guarantee that new knowledge will actually be created, but it increases the chances that such creation will take place.

Thus, aside from creating new knowledge assets through abstraction, codification, impacting and absorption, another way in which new assets are created in the simulation model is when the Linkage Probabilities between (a) a Node and a Link of equal Complexity; and (b) that same Link with another Node of equal Complexity, are sufficiently high. When the Linkage Probabilities in such a Node-Link-Node chain exceeds a specified threshold, a new Asset is created. Linkage Probability thus measures the propensity for an Asset to combine with Assets of the same level of Complexity but of the complementary type – i.e., nodes with links, and links with nodes. Every possible Node-Link pair of the same level of complexity has some linkage probability even if this turns out to be zero.

Each Agent that owns the necessary constituent Assets and that has made the necessary investment, will discover the new Asset once the appropriate probability threshold has been crossed. The resulting asset will necessarily be more complex than its constituent elements. In the model, the new Asset will have a level of Complexity that is 1 notch further up on the complexity scale than its constituent Assets.[4]

Background Processes. Two background processes, Obsolescence decay and Diffusion decay, shape the environment in which the simulation takes place:

Obsolescence Decay: Obsolescence decay measures the loss of value of existing knowledge assets due to their obsolescence over time. Obsolescence is represented as the shrinking of the Rents Multiplier of existing Assets over time. The rate of decrease is set as a function of the industry being simulated.

Diffusion Decay: Diffusion decay is the loss of value of existing knowledge assets due to the unintended diffusion – the 'spillovers' - of these assets.This is represented as the random distribution of Assets to a greater number of Agents both within and outside the model. Each Agent coming into the possession of these new Assets will make an independent decision as to whether it will incorporate them into its Active or Passive sets, or whether it will ignore them (if the agent is fully laden). The probability and rate at which a given Asset will diffuse will be a function of:

- Its degree of abstraction and codification, in accordance with the tenets of the I-Space, and
- An industry-specific factor.

Blocking Diffusion Decay: Blocking diffusion decay is in the hands of an agent. Agents are initialized to block or not to block diffusion decay. Investments in diffusion decay will be a percentage cost of the rents potential of the asset and proportional to diffusion.

[4] Note that only Assets of equal Complexity have any Linkage Probability, thus all constituent Assets will have the same level of Complexity.

3.3 The Components of Agent Interactions

Agent Meetings: Agent meetings can be either arranged or random.

- *Arranged meetings*: At the beginning of each period, Agents will identify Agents they wish to meet. Arranging to meet an Agent imposes an Arrangement cost. Arranged meetings can occur between any numbers of Agents, and only take place when all parties involved arrange to meet all the other Agents involved. I.e. a 3-party meeting only takes place when all 3 participating Agents choose to meet the other 2 Agents in the same period.
- *Random meetings*: Aside from arranged meetings, random meetings can occur between any two Agents. Random meetings do not incur any arrangement costs. The users of the simulation are free to specify the frequency with which agents will meet in each period.

Meeting Costs: Agent meetings impose four different kinds of costs on participants – arrangement costs, fixed costs, presentation costs and inspection costs. *All* meeting costs are Financial Costs. They are set against the Financial Budget allocated for the period and thus consume Financial Funds.

- *Arrangement cost*: The arrangement cost is levied on the Agent for each Agent he attempts to arrange a meeting with – regardless of whether a meeting results from the attempt. This cost only applies to attempts to arrange meetings and does not apply when meetings are random. The arrangement cost of a meeting does not vary as a function of the history of past transactions.
- *Fixed cost*: The fixed cost of a meeting is levied on all participants of a meeting, regardless of size.[5] The fixed cost of a meeting does not vary as a function of the history of past transactions.
- *Presentation costs*: Presentation costs are the costs of making available the agent's Trading Set for inspection. They are imposed once for each meeting that the agent attends.[6] Presentation costs increase with the number of knowledge assets that the agent presents, and decrease both with the degree of abstraction and codification of these assets—in line with I-Space thinking—as well as with the degree of recurrence of transactions with a given agent or group of agents.
- *Inspection cost*: The inspection cost is the cost of an Agent inspecting the Trading Set that has been offered for its examination, and is levied once for each Trading Set that the Agent inspects. Each Trading Set

[5] A benefit of a large meeting is that Agents amortize the fixed cost over a greater number of potential transactional partners.

[6] A second benefit of a large meeting is that Agents amortize the presentation cost over a greater number of potential transaction partners.

imposes an Inspection cost that increases with the number of Assets in the Trading Set, and decreases with the Abstraction and Codification of those Assets,[7] as well as decreasing with recurrence.

Transactional options: Agent meetings can result in the following transactions taking place between agents:

- *The trading of knowledge assets*: In a trade, agents exchange knowledge assets for Financial Funds. There is no change in the level of diffusion of the assets traded since the selling agent relinquishes all rights to the assets in exchange for a stock of Financial Funds received.
- *The licensing of knowledge assets*: In licensing, agents share their knowledge assets in return for Financial Funds. Nominal Diffusion (see above) increases since the agent who grants the license to the Asset retains its rights to use it. In return for the right to use the Asset, the agent receiving the license (the licensee) pays the agent granting the license (the licensor) a continuous flow of Financial Funds.
- *Joint Ventures*: In a joint venture, agents come together to create a new agent that is jointly owned. The new agent receives an injection of Financial Funds, Experience Funds and Assets from its "parent" agents. The parents continue to exist as independent agents while receiving a variable flow of Financial Funds from the Agent created as a result of the joint venture. The flow will be proportional both to the rents and to their respective investments in the joint venture.
- *Mergers*: In a merger, Agents come together to create a new Agent, by pooling all their Assets, Financial Funds and Experience Funds. The original Agents cease to exist as independent Agents and will instead be represented by this new Agent. Thus, in a merger, the total number of agents in the simulation actually decreases.
- *The creation of subsidiaries*: In creating a subsidiary, one agent unilaterally creates a new Agent, which receives an injection of Financial Funds, Experience Funds and Assets from the "parent" Agent. The original Agent continues to exist independently, and earns a variable flow of Financial Funds dependent on the success of the subsidiary.[8]

While all transactions can take place regardless of the number of participants at a meeting, all transactions are bilateral transactions. I.e. In a meeting with five Agents, the decision to trade an Asset still takes place between two participants who have presented and inspected each other's Trading Set -

[7] Inspection cost forms the limiting factor to the size of multi-agent meetings. As the number of potential transaction partners increase, the cost of attending such a meeting increases.

[8] An Agent that finds itself with far too many unrelated Assets may carve off portions of it into subsidiaries.

ditto for licensing, joint ventures and mergers. Thus meetings with more than two Agents serve the primary function of efficiently bringing more Agents together, but do not, of themselves, constitute new transactional options. The situation is summarized in Table 2.

Table 2. Multi-agent meetings & 2-agent meetings

	N-Party Meeting (Arranged)	**2-Party Meeting (Arranged/Random)**
Trading Set Presented	Based on lowest degree of recurrence with all (N-1) other Agents.	Based on degree of recurrence with the other participating Agent.
Arrangement Costs	Pays arrangement cost for (N-1) Agents that each Agent elects to meet.	If the meeting is arranged, the Agent pays the arrangement cost.
Presentation Costs	Each Agent presents once at each meeting, and pays the presentation cost for the Trading Set it presents once for each meeting it attends.	
Inspection Costs	Each Agent inspects (N-1) Trading Sets, and pays the associated inspection costs	Each Agent inspects the other's Trading Set, and pays the associated inspection costs
Fixed Costs	Each Agent pays the Fixed cost of a Meeting once for each Meeting it attends.	

4 Acknowledgements

The authors would like to thank Wharton-SMU Research Center of Singapore Management University for their financial support in carrying out this research.

A Description of Variables

We offer a summary description of the Sim-I-Space variables below together with their settings for a typical run.

A.1 Input Parameters for Global Switches

ASSET_MANAGEMENT

- This activates the asset management module.
- When it is a true, mod_discovery and mod_research methods will run, if either of them is true.

MOD_DISCOVERY

- In the I-space world each agent discovers Nodes and Links to generate new Nodes or Links, after checking the linkage probability between Nodes and a Link.

- If the Mode of Discovery sets to 0 (TRUE), then the 'generateDiscovery' Method is activated.

Otherwise, it will be disabled.

MOD_RESEARCH

- Each agent moves a Node or a Link to create new Node or Link in the I-space.
- If the Mode of Research sets to 0 (TRUE), then the 'generateResearch' Method is activated.

Otherwise, it will be disabled.

AGENT_INTERACTION

- The module that activates encounters and interactions among agents can be switched on or off.
- If the AGENT_INTERACTION sets to 0 (TRUE), then the 'generate-Meetings' Method is activated.

Otherwise, it will be disabled.

MOD_OBSOLESCENCE

- This constitutes an end-of-Period functions.
- The module for generating Obsolescence Decay for each node or link can be switched on or off.
- If the MOD_OBSOLESCENCE sets to 0 (TRUE), then the 'generateObsolescenceDecay' Method is activated.

Otherwise, it will be disabled.

MOD_DIFFUSION

- This constitutes an end-of-Period functions.
- The module for generating Diffusion Decay for each node or link can be switched on or off.
- If the MOD_DIFFUSION sets to 0 (TRUE), then the 'generateDiffusion-Decay' Method is activated.

Otherwise, it will be disabled.

A.2 Input Parameters for Main Variables

INIT_AGENT_NUM

- Number of Agents at the start of the simulation
- What an Agent is depends on the nature of the simulation - it could be a firm, an employee within a firm, or something intermediate like an SBU or a department within a firm. The key requirement is that Agents be endowed with both data-processing and decision-making powers and that they be homogeneous with respect to the attribute that defines them as an Agent.
- Possible range (current): 1 - 50 (20)

INIT_NODE_NUM

- Number of distinct knowledge Nodes at the start of the simulation
- The initial number of Nodes will be distributed randomly to individual Agents
- Knowledge Nodes are identifiable pieces of knowledge that can stand on their own when generating revenue. They can themselves be made up of nested networks of knowledge Nodes and Links
- Possible range (current): 1 - 50 (25)

INIT_LINK_NUM

- Number of distinct knowledge Links at the start of the simulation
- The initial number of Links will be distributed randomly to individual Agents
- Knowledge Links establish relationships between knowledge Nodes. They constitute identifiable pieces of knowledge that help to integrate knowledge assets together.
- Possible range (current): 1 - 30 (15)

MODEL_PERIODS

- Number of periods that the simulation will run for
- Possible range (current): 1 - 200 (200)

MAX_AGENT_NUM

- Maximum number of Agents allowed during the simulation
- Possible range (current): 50 - 300 (50)

MAX_NODE_NUM

- Maximum number of distinct knowledge Nodes allowed during the simulation
- Possible range (current): 100 - 5000 (300)

MAX_LINK_NUM

- Maximum number of distinct knowledge Links allowed during the simulation
- Possible range (current): 100 - 3000 (200)

A.3 Input Parameters for Asset Variables

MAX_COMPLEXITY

- Maximum level of Complexity of Assets allowed in the simulation
- Derived from the "maximum" levels of Complexity perceivable by agents in the simulation.
- Possible range (current): 1 - 10 (7)

COMPLEXITY_COST

- Rate at which holding Assets of increasing Complexity increase their carrying cost
- Derived from MAX_COMPLEXITY (see above) such that MAX_COMPLEXITY is the level at which a fully Abstract, Codified and Diffused knowledge Asset remains economically viable.
- Possible range (current): 0.00 - 1.00 (0.001)

NEW_LINK_PROB

- The probability that a new Asset generated by combining two Nodes and a Link is a Link
- When Agents research the creation of a new Asset via the combination of existing Nodes/Link, the resulting new Asset can be either a Node or a Link, and NEW_LINK_PROB determines the ratio of Nodes to Links among new Assets.
- Possible range (current): 0.00 - 0.50 (0.20)

MOVE_POSSIBILITIES

- Number of possibilities for each type of movement in I-Space for each Asset
- Movement in I-Space creates new knowledge Assets, and for each existing knowledge Asset there is a finite number of possible new Assets locations available in I-Space.
- Possible range (current): 1 - 5 (2)

PASSIVE_CARRY_MULT

- The carrying cost of knowledge Assets in the Passive set relative to that of knowledge Assets in the Active set
- Knowledge Assets in the Passive set cost less to maintain as they are not being actively managed in the I-Space to generate Revenue.
- Possible range (current): 0.00 - 1.00 (0.50)

A.4 Input Parameters for I-Space World Variables

BASE_REV_MULT

- A multiplier that determines the allocation of Base Revenue Potential within the I-Space
- Base Revenue Potential in the I-Space is proportional to the degree of Abstraction and Codification, and is inversely proportional to the degree of Diffusion.
- Possible range (current): 0.00 - 2.00 (1.00)

ASSET_SHARE_PROB

- The probability that a given Agent at the start of the simulation gets initially allocated a given Asset.
- The initial portfolio of Assets is distributed among such Agents based on this probability. Each Agent will have
  ASSET_SHARE_PROB probability of getting each Asset. This is a measure of the initial homogeneity knowledge in the simulation at the beginning of the game.
- Possible range (current): 0.00 - 1.00 (0.20)

OBSOLESCENCE_DECAY

- Factor controlling the rate at which Assets become obsolete over the course of the simulation
- The rate at which the Revenue Multiplier of Assets decays per period due to obsolescence is proportional both to OBSOLESCENCE_DECAY and the total number of new knowledge Assets that have been created in that period. What is being measured here is the "creative destruction" that goes on within an industry. Dynamic fast-moving industries are subject to much higher rates of obsolescence decay than slower moving ones.
- Possible range (current): 0.00 - 0.50 (0.001)

DIFFUSION_DECAY

- Factor controlling the rate at which Assets are diffused
- The rate at which Assets diffuse to a broader population each period is proportional to DIFFUSION_DECAY and the degree of Abstraction and Codification of the Asset
- There are two forms of Diffusion Decay: 1) diffusion of Assets to Agents *within* the simulation increases the degree of Diffusion of Assets - as measured by the simulation metrics; 2) diffusion of Assets to a population *outside* the simulation does not increase the degree of Diffusion of the Asset - as measured by the simulation metrics - but decreases the Revenue Multiplier of the Asset directly.
- Possible range (current): 0.00 - 0.05 (0.001)

DIFFBLOCK_COST_MULT

- This is a factor controlling the rate at which the cost of blocking the diffusion of assets is set.
- The formula for calculating the cost of blocking the diffusion is as follows: The diffusion block cost = DIFFBLOCK_COST_MULT * Abstract * Codify * Diffuse.
- Possible range (current): 0.00 - 0.20 (0.001)

AGENT_ENTRY_THRESHOLD

- Threshold of mean Agent revenues above which new Agents are attracted into the simulation
- The number of new Agents is based on an estimate of the number of Agents the current "market" can support
- Possible range (current): 1.00 - 5.00 (1.25)

AGENT_ENTRY_RATE

- The Agent Entry Rate is the number Agents entering per percent change in revenues between this period and the last period.
- For example, setting AGENT_ENTRY_RATE to 0.25 means that for every percent increase in revenues, 0.25 Agents will be attracted into the game, translating into a simple number of a 1 agent for every 4% increase this period. N.B. Although this would be unlikely, one could conceivably have a negative Agent Entry Rate, i.e. at -0.25, this means that 1 agent is attracted into the game for every 4% decrease during this period.
- Possible range (current): 0.01 - 1.00 (0.25)

AGENT_EXIT_THRESHOLD

- Agents exit is controlled by two variables - an Agent Exit Threshold and an Agent Exit Probability variable. Thus an Agent may decide to exit depending on how much financial funds it has managed to amass, and depending on the current trend of its revenues.

- The Agent Exit Threshold determines the current level of financial funds that the Agent must achieve before it will consider exiting. This in essence sets the level at which an Agent feels that it is sufficiently "ahead" of the game to choose to quit while it is ahead.
- Possible range (current): 5.00 - 50.00 (12.50)

AGENT_EXIT_PROB

- The Agent Exit Probability is the probability of any one Agent exiting per % change in that Agent's revenues between this period and the last period. This function works in both directions, depending on the value – i.e., setting AGENT_EXIT_PROB to 0.01 means that for every % increase in revenues, each Agent has a 0.01 (or 1%) chance of deciding to exit.

Similarly, setting this variable to -0.02 means that for every % decrease in revenues, each Agent has a 0.02 (2% chance of deciding exit). Thus by playing with the positive and negative values for this, you can have Agents that decide to leave when revenues are increasing, or Agents that leave when revenues are declining.

- Possible range (current): -0.007 $\sim$ +0.007 (-0.005)

MAX_AGENT_ENTRIES

- Maximum number of new Agents that will enter per period
- Possible range (current): 1 - 10 (5)

A.5 Input Variables for I-Space Matrix Variables

ABSTRACT_DIM_SIZE

- Size of the Abstraction dimension
- This establishes the number of discrete intervals in the Abstraction dimension
- Possible range (current): 2 - 10 (5)

CODIFY_DIM_SIZE

- Size of the Codification dimension
- This establishes the number of discrete intervals in the Codification dimension
- Possible range (current): 2 - 10 (5)

DIFFUSE_DIM_SIZE

- Size of the Diffusion dimension
- This establishes the number of discrete intervals in the Diffusion dimension
- Possible range (current): 2 - 10 (4)

DIFFUSE_FACTOR

- Factor which determines rate at which model Diffusion increases with increases in nominal Diffusion (see text for further details)
- This variable establishes how to convert nominal Diffusion (the number of Agents in the simulation that own the Asset) into model Diffusion (the measure of Diffusion which directly impacts the location of the Asset in discrete I-Space. The DIFFUSE_FACTOR essentially converts a linear discrete model Diffusion into a logarithmic scale.
- Possible range (current): 2 - 5 (2)

A.6 Input Variables for Linkage Probability Variables

ABS_NONZERO_VAL

- The value that is given by increasing abstraction to particular cell in the Linkage Probability matrix – prior to increasing abstraction its value was set at zero.
- Possible range (current): 0.00 to 0.50 (0.20)

COD_INCREASE_VAL

- The increase in the Linkage Probability value that occurs following an increase in codification in a cell with a prior non-zero value.
- Possible range (current): 0.00 to 0.50 (0.20)

ABS_NONZERO_PROB

- The degree to which an increase in the degree of Abstraction translates into an increase in the number of non-zero linkage probability entries in the Linkage Probability matrix
- Abstraction is measured by the number of non-zero entries in the respective rows or columns in the Linkage Probability matrix. The larger the number of non-zero entries, the wider the range of potential linkages – ie, applications - a given Node or Link has.
- As increasing Abstraction increases the number of non-zero linkage probability entries in the Linkage Probability matrix, so decreasing Abstraction increases the number of zero linkage probability entries in the matrix.
- Possible range (current): 0.00 to 0.25 (0.05)

NEW_ASSET_THRESHOLD

- The complexity threshold beyond which a given Node-Link-Node chain creates a new knowledge Asset of greater Complexity.
- Possible range (current): 0.50 to 1.00 (0.50)

NEW_ASSET_PROB

- Base probability of successfully achieving the new Asset combination made possible by a given Node-Link-Node chain whose Linkage Probability values exceed NEW_ASSET_THRESHOLD
- Possible range (current): 0.00 - 3.00 (1.00)

A.7 Input Parameters for Agent Variables

INIT_FINANCIAL_FUNDS

- The initial endowment of Financial Funds for initial Agents
- Possible range (current): 1.0 - 50.0 (10.0)

INIT_EXPERIENCE_FUNDS

- The initial endowment of Experience Funds for initial Agents
- Possible range (current): 1.0 - 50.0 (10.0)

FINANCIAL_ALLOCATION

- The ratio of financial funds allocated from revenue and costs.
- For example, 0.2 means that 20% of the revenue (cost) is allocated to financial funds (expense).
- Possible range (current): 0.0 - 1.0 (0.5)

ACTIVE_SET

- The size of the Active set
- Possible range (current): 1 - 50 (10)

PASSIVE_SET

- The size of the Passive set
- Possible range (current): 1 - 50 (10)

ISPACE_MOVE_PROB

- The probability of successfully moving an Asset inside I-Space
- The probability of successfully moving an Asset in I-Space is proportional to the Asset's Abstraction and Codification.
- Possible range (current): 0.01 - 0.10 (0.03)

ISPACE_MOVE_COST

- A factor which determines the cost of moving an Asset in I-Space
- The cost of moving an Asset in I-Space is proportional to the Asset's Abstraction and Codification.
- Possible range (current): 0.01 - 0.10 (0.03)

JOINT_VENTURE_RETURN

- The proportion of a Joint Venture Agent's net income that is returned to parent Agents
- Possible range (current): 0.01 - 1.00 (0.10)

SUBSIDIARY_RETURN

- The proportion of a Subsidiary Agent's net income that is returned to parent Agents
- Possible range (current): 0.01 - 1.00 (0.20)

PAR_FUND_THRESHOLD

- The financial fund threshold required of a parent company that allows it to create a subsidiary company
- Possible range (current): 1 - 1000 (15)

PAR_ASSET_THRESHOLD

- The asset threshold required of a parent company that allows it to create a subsidiary company
- Possible range (current): 0.01 - 1.00 (0.80)

A.8 Input Parameters for Agent Meeting Variables

MEETING_ARRANGE_COST

- The cost of attempting to schedule a meeting with an Agent
- Possible range (current): 0.00 - 0.10 (0.005)

MEETING_FIXED_COST

- The fixed cost of attending a meeting - irrespective of size
- Possible range (current): 0.00 - 0.10 (0.005)

PRESENT_COST_MULT

- The factor that sets the cost of presenting an Asset at a meeting
- The cost of presenting an Asset is proportional to the Asset's Abstraction and Codification.
- Possible range (current): 0.01 - 0.10 (0.005)

EXAMINE_COST_MULT

- The factor that sets the cost of inspecting an Asset at a meeting
- The cost of inspecting an Asset is proportional to the Asset's Abstraction and Codification.
- Possible range (current): 0.01 - 0.10 (0.005)

TRADE_VALUE_MULT

- The multiplier that is used to determine the tradeable value of an Asset
- Possible range (current): 1.00 - 5.00 (1.75)

LICENSE_VALUE _MULT

- The multiplier that is used to determine the licensing value of an Asset
- Possible range (current): 0.01 - 2.00 (0.25)

A.9 Input Parameters for Meetingspace Variables

PROB_RANDOM_MEETINGS

- The probability of random meetings per Agent per period
- Possible range (current): 0.0 – 1.0 (0.25)

MAX_RANDOM_MEETINGS

- The maximum number of random meetings per Agent per period
- Possible range (current): 0 - 10 (5)

TRADING_SET_RATIO

- The factor that is used to determine size of the Trading Set that an agent will deploy. This is based on the number of prior meetings that the focal agent has had with a given Agent
- Possible range (current): 1 - 5 (2)

A.10 Input Parameters for DM Variables

MIN_PREFERENCE_LEVEL

- The minimum preference that a given agent can display for abstraction, codification, impacting, and absorption.
- Possible range (current): 0.0 – 1 (1)

MAX_ PREFERENCE_LEVEL

- The maximum preference that a given agent can display for abstraction, codification, impacting, and absorption.
- Possible range (current): 0 - 10 (5)

A.11 Input Parameters for Research DM Variables

PROB_MOVE

- The probability that an Agent will attempt to move a given knowledge Asset in I-Space
- Possible range (current): 0.00 - 1.00 (0.20)

A.12 Input Parameters for Meeting DM Variables

PROB_POSITIVE

- The base probability of a focal agent adopting a positive disposition towards a given Agent
- Possible range (current): 0.00 - 0.50 (0.10)

PROB_NEGATIVE

- The base probability of a focal agent adopting a negative disposition towards a given Agent
- The probability of having a neutral disposition is thus 100% - PROB_POSITIVE - PROB_NEGATIVE.
- Possible range (current): 0.00 - 0.50 (0.20)

HIGH_COOP_MULT

- The multiplier used to determine the level of cooperation among agents.
- Possible range (current): 0.00 – 0.10 (0.50)

PROB_TRADE (UNUSED)

- The probability of a given agent offering a Trade during a meeting with another Agent
- Possible range (current): 0.00 - 1.00 (0.25)

PROB_VARIABILITY (UNUSED)

- Possible range (current): 0.00 - 1.00 (0.05)

JOINT_VENTURE_INVEST

- The rate of the investment for a joint venture
- Possible range (current): 0.00 - 1.00 (0.25)

SUBSIDIARY_INVEST

- The rate of the investment for a subsidiary
- Possible range (current): 0.00 - 1.00 (0.25)

B Detailed Model Specification with Example

B.1 Structure of the Sim-I-Space Model

1. **Assets**
   (a) **Asset ID and Type**: Knowledge Assets ("Assets") are either Nodes or Links. Each Asset is identified by a unique ID number that is allocated sequentially, i.e. no two Assets (Node or Link) will have the same ID, and the first Asset will have ID 0, and the next Asset will have ID 1, and so on.
   (b) **Abstraction and Codification**: Each Asset has discrete attributes of Abstraction and Codification, which represent the degree each Asset is abstract or codified respectively. Values for Abstraction and Codification begin at 0 (respectively representing concrete and uncodified Assets), and have an upper bound that is specified in the model to give varying degrees of granularity.
   (c) **Diffusion**: The attribute of Diffusion represents the degree each Asset is diffused among the Agents in the model. The effect of Diffusion on the value of the Asset is non-linear – the loss in value from one additional Agent coming into possession of the Asset varies greatly depending on whether there was originally only one Agent who owned the Asset, or if there were 100 Agents who owned the Asset. In the former case, there is a sharp loss in value, and in the latter case, there may be almost no loss in value. To reconcile this difference there are two relevant values of Diffusion.
      i. **Nominal Diffusion**: Nominal Diffusion is a continuous variable that measures the actual number of Agents that own the Asset. Nominal Diffusion has no upper bound.
      ii. **Model Diffusion**: Model Diffusion is a discrete variable derived from nominal Diffusion that is used in the I-Space Model to calculate the Base Revenue Potential of the Asset. Model Diffusion has an upper bound specified in the model to give varying degrees of granularity.

   For example, in a model with model Diffusion values of 0∼3, model Diffusion may be determined as in Table 3.

Table 3. Nominal diffusion & model diffusion

Model Diffusion	**Nominal Diffusion (# of Agents that own the Asset)**
0	$1 \sim 2$
1	$3 \sim 8$
2	$9 \sim 26$
3	$27 +$

In this example, there is no difference in the Base Revenue Potential if 1 or 2 Agents own the Asset, but when a 3^{rd} Agent owns the Asset, the Base Revenue Potential decreases. Base Revenue Potential then remains steady until the Asset diffuses to the 9^{th} Agent, etc.

(d) **Complexity**: Complexity describes the complexity of the knowledge structure represented by the Asset. Complexity begins at 0 (basic Assets) and has no upper bound. Assets combine with Assets of equal Complexity, and the resulting Asset is of one larger Complexity.

(e) **Revenue Multiplier**: The Revenue Multiplier is the link between Base Revenue Potential (a function of where the Asset is in the I-Space) and actual revenue earned per unit time. The Revenue Multiplier accounts for a variety of factors, including Complexity (increasing as Complexity increases[9]), obsolescence (decreasing over time subject to the Obsolescence decay function), and "jackpot effects" (random extraordinary increases to the payout from combining Assets).

(f) **Carry Cost**: Maintaining Assets impose a Carry Cost on Agents. The Carry Cost is a function of the Asset's Complexity[10].

(g) **Node/Link Parents**: New Assets can be created as a result of the process of abstraction, codification, impacting and absorption, or by the combination of a chain of Node/Link/Node of sufficiently high Linkage Probability. The "parents" of an Asset are thus Nodes and/or Link from which this Asset was derived from. For initial/basic Assets, they have no "parents".

(h) **Possible new Assets due to "movement"**: The processes of abstraction, codification, impacting and absorption yield new Assets. However, there are only a finite number of ways in which any given Asset can be abstracted, codified, impacted or absorbed – thus each Asset maintains memory of what are the possible results of abstracting, codifying, impacting or absorbing itself. Furthermore, the process of abstraction, codification, impacting and absorption is idiosyncratic – thus some of the results will be more common than others. E.g., it may be that the absorption of Node 3, will yield only Node 6, Node 9 or Node 23. However, 50% of the Agents who succesfully absorb Node 3 will discover Node 6, 33% will discover Node 9, and the remaining 17% discover Node 23.

2. **I-Space Matrix and I-Space Locations**

(a) **I-Space Matrix and I-Space Locations**: The I-Space Matrix is a three-dimensional matrix representation of the I-Space, with axes for

[9] By combining Assets, Agents create new Assets of higher Complexity (and Revenue Multiplier). By doing so, Agents can offset Carry Costs, and usage of Agent memory (by replacing original Assets with the new combined Asset), while retaining some of the value of the original Assets as revenue generators.

[10] Increasing Complexity thus has two conflicting effects – increasing both the Revenue Multiplier, and the Carry Cost.

Abstraction, Codification and Diffusion. The I-Space Matrix is made up of I-Space Locations - discrete locations in the I-Space Matrix where Assets reside. Each I-Space Location is uniquely described by discrete values of Abstraction, Codification and Diffusion, and can contain any number of Assets. Each I-Space Location in the I-Space Matrix is associated with some Base Revenue Potential.

(b) **Base Revenue Potential**: The Base Revenue Potential reflects the fundamental relationship between Revenue and the three attributes of Abstraction, Codification and Diffusion. Thus, each I-Space Location in the I-Space Matrix is associated with a Base Revenue Potential that is used for all Assets residing in that location. The Base Revenue Potential is further subject to an Industry Multiplier that determines the difference between the Base Revenue Potential of two Assets of equal Abstraction, Codification, and Diffusion in two different industries.[11]

3. **Linkage Probability Matrix**
   (a) **Linkage Probability**: Linkage Probability measures the affinity for an Asset to combine with Assets of the same Complexity, but of the alternate type (Nodes with Links, and Links with Nodes). Every possible Node-Link pair with the same Complexity has some Linkage Probability (although this Linkage Probability may be zero).
   (b) **Linkage Probability Matrix**: The Linkage Probability Matrix is a two-dimensional matrix that exists for each level of Complexity. Each column in the matrix represents one of every existing Node of the given level of Complexity, and each row a Link. Thus the entry in row i, and column j, is the Linkage Probability between the Link in row i, and the Node in column j.
4. **Agent**
   (a) **Agent ID**: Agents in the model are identified by a unique ID that is allocated sequentially, i.e. no two Agents will have the same ID, and Agent 0 was the first Agent created, while Agent 1 was the next, and so on.
   (b) **Financial Funds and Experience Funds**: Agents possess resources that are used to cover their operating expenses. These resources, and associated expenses, fall into two categories – Financial and Experience. Agents manage their resources by allocating revenues into either Financial or Experience funds, and setting Financial and Experience Budgets.
      i. **Financial Funds and Financial Budget**: Financial Funds correspond tangible resources, such as cash, that are used to meet operating expenses, including the cost of meeting other Agents,

[11] Base Revenue Potential should not be confused with Revenue Potential - the former is invariant over time in each simulation run, while the latter is the realized revenue per period generated by each Asset, varying over time.

trading for Assets, and paying out dividends. The Financial Budget is set by the Agent each period to limit the amount of the Financial Funds that the Agent intends to expend each period.

ii. **Experience Funds and Experience Budget**: Experience Funds correspond to more intangible resources, such as the experience from learning-by-doing, that are used to manipulate Assets – to increase/decrease their Abstraction or Codification, or to combine Assets to create new Assets. The Experience Budget is set by the Agent as a limit on the amount of the Experience Funds that the Agent is intending to expend each period.

(c) **Active set and Passive set**: Agents have a finite memory that is separated into two sets – an Active set and a Passive set. Assets can be held in either of these sets, although storing Assets invoke Carry Costs. Agents manage their memory by choosing to hold Assets in either of the two sets, or discarding them altogether.

i. **Active set**: The Active set contains all Assets being actively utilized by the Agent, and generating revenue.

ii. **Passive set**: The Passive set contains all Assets that the Agent possesses, but are not utilized, and not generating revenue. Maintaining Assets in the Passive set alleviate some of the Carry Cost associated with the Assets.

(d) **Trading Set**: Trading Sets are sets of Assets that an Agent makes available for transactions with other Agents. When two Agents meet, not all of their Assets may be available for transactions. In the first meeting, the cost of presenting Assets will limit the sharing to only the most Abstract and Codified Assets. Over time, as the degree of familiarity between two Agents increase with recurrent meetings, the cost of presenting Assets will decrease, and Agents will be able to share more concrete and uncodified Assets (thus increasing the size of the set of Assets available for transactions). Agents thus have more than one Trading Set, and it presents different Trading Sets to different Agents it meets depending on the degree of familiarity (i.e. history of meetings).

(e) **Agent Memory**: Agent memory stores the frequency of the historical encounters with other Agents.

B.2 Asset Evolution in the Model

5. **Movement/Creation of Assets**: The processes of abstraction, codification, impacting and absorption are the processes by which Assets evolve within I-Space.
   (a) **Abstraction**: Abstraction of an Asset creates a new Asset of the same type and Complexity with the next higher Abstraction value. The new Asset inherits the Linkage Probabilities of its predecessor, with an increase in the number Assets it can link with, i.e., an increase in the number of non-zero values in the corresponding row (for Links), or column (for Nodes) in the Linkage Probability matrix.
   (b) **Codification**: Codification of an Asset creates a new Asset of the same type and Complexity with the next higher Codification value. The new Asset inherits the Linkage Probabilities of its predecessor, with an increase in its ability to link with Assets, i.e., an increase in existing non-zero values in the corresponding row (for Links) or column (for Nodes) in the Linkage Probability matrix.
   (c) **Impacting**: Impacting of an Asset, a new Asset of the same type and Complexity with the next lower Abstraction value. The new Asset inherits the original Linkage Probabilities of its predecessor, with a decrease in the number Assets it can link with, i.e., a decrease in the number of non-zero values in the corresponding row (for Links), or column (for Nodes) in the Linkage Probability matrix.
   (d) **Absorption**: Absorption of an Asset creates a new Asset of the same type and Complexity with the next lower Codification value. The new Asset inherits the Linkage Probabilities of its predecessor, with a decrease in its ability to link with Assets, i.e. a decrease in existing non-zero values in the corresponding row (for Links) or column (for Nodes) in the Linkage Probability matrix.
6. **Creation of new Assets:** Aside from abstraction, codification, impacting and absorption, another way in which new Assets are created in the model is when the Linkage Probabilities between a Node (N_x) and a Link (L_y) of equal Complexity; and that same Link (L_y) and another Node (N_z) of equal Complexity are sufficiently high. When the Linkage Probabilities in such a Node-Link-Node chain exceeds a specified threshold, an Agent can attempt to research the creation of a new Asset.
   (a) **New Asset:** The result of combining a given set of Node-Link-Node can either be a Node or a Link. However, the result of combination is unique (?), i.e., every Agent that owns the necessary constituent Assets that successfully research the creation of a new Asset from this will discover the same new Asset. The new Asset is also necessarily more complex than its constituents, and in the model, the new Asset has Complexity 1 larger than the Complexity of the constituent Assets.[12]

[12] Note that only Assets of equal Complexity have any Linkage Probability, thus all constituent Assets will have the same Complexity.

B.3 Agent Interaction in the Model

7. **Agent Meetings**: Agent meetings can take place between two Agents (bilateral) or between a larger number of Agents (multilateral). To ensure that meetings remain manageable, there is a cap on the largest possible size for a multilateral meeting. In order for a meeting to occur, participants in a meeting must have either arranged to meet the other Agent(s), or have met the Agent(s) in a random encounter.
   (a) **Arranged Meetings**: At the beginning of each period, Agents will define their disposition towards other Agents in the game. This disposition can either be:
      i. strong desire for meeting – an Agent who registers such a disposition will actively seek to arrange a meeting with the subject Agent, and will agree to a meeting arranged by the subject Agent;
      ii. amenable to meeting – an Agent with this disposition will not seek to arrange a meeting with the subject Agent, but will agree to a meeting arranged by the subject Agent; or
      iii. no desire for a meeting – an Agent with this disposition will neither seek to arrange a meeting with the subject Agent, nor will it be likely to agree to a meeting arranged by the subject Agent.
   (b) **Random Meetings**: Aside from arranged meetings, random meetings can occur between any two Agents. Random meetings do not incur any cost to arrange, but Agents do not have control over the Agents they encounter.

   Based on the result of the attempts to arrange meetings, as well as the random encounters in each period, Agent meetings are generated.
   In the model, Agents are aware of the economies of scale present in multi-Agent meetings, and thus have a default preference for attending the largest possible multi-Agent meetings (meetings with more than 2 Agents), but are however limited to attending only one multi-Agent meeting each period.
8. **Presentation and Inspection:** Agents attending meetings will present their Trading Set for inspection, as well as inspecting the Trading Set of other Agents attending the meeting. In a multi-Agent meeting, an Agent will present only one Trading Set. However, as it will have varying degrees of familiarity with the other Agents attending the meeting, it will restrict itself to the smallest Trading Set among all the Trading Sets it would present if it had met all the participants in bilateral transaction. E.g., if Agent X attends a meeting with Agent Y and Agent Z, Agents that it has met 9 times and 4 times before respectively, the Trading Set Agent X presents to the group will be the same as the Trading Set it would have presented to Agent Z (the Agent with which it has the lowest degree of familiarity).

9. **Meeting Costs**: Agent meetings impose four different kind of costs on participants – arrangement costs, fixed costs, presentation costs and inspection costs. All meeting costs are Financial Costs – they are set against the Financial Budget allocated for the period and consume Financial Funds.
   (a) **Arrangement cost**: The arrangement cost is levied on the Agent for each Agent he attempts to arrange a meeting with – regardless of whether a meeting results from the attempt. This cost only applies to attempts to arrange meetings and does not apply when meetings are random. The arrangement cost of a meeting is invariant regardless of the history of past transactions.
   (b) **Fixed cost**: The fixed cost of a meeting is levied on all participants of a meeting, regardless of size.[13] The fixed cost of a meeting is invariant regardless of the history of past transactions.
   (c) **Presentation cost**: The presentation cost is the cost of making available the Agent's Trading Set for inspection, and is levied once for each meeting that the Agent attends.[14] Presentation cost increases with the number of Assets the Agent presents, and decreases with the Abstraction and Codification of the Assets (i.e. varies depending on the Trading set presented), as well as decreasing with recurrence.
   (d) **Inspection cost**: The inspection cost is the cost of an Agent inspecting the Trading Set that has been offered for its examination, and is levied once for each Trading Set that the Agent inspects. Each Trading Set imposes an Inspection cost that increases with the number of Assets in the Trading Set, and decreases with the Abstraction and Codification of those Assets,[15] as well as decreasing with recurrence.
10. **Meeting Transactions:** The following transactions can occur in an Agent meeting:
   (a) **Trading of Assets**: Trading is a bilateral transaction in which Agents exchange Assets for Financial Funds. There is no change in Diffusion as the selling Agent relinquishes all rights to the Assets in exchange for a stock of Financial Funds received.
   (b) **Licensing of Assets**: Licensing is a bilateral transaction where Agents share Assets for Financial Funds. Nominal Diffusion increases as the Agent who grants the license to the Asset retains its rights to use the Asset. In return for the right to use the Asset, the Agent receiving the license pays the first Agent a continuous flow of Financial Funds.

[13] A benefit of a large meeting is that Agents amortize the fixed cost over a greater number of potential transactional partners.

[14] A second benefit of a large meeting is that Agents amortize the presentation cost over a greater number of potential transaction partners.

[15] Inspection cost forms the limiting factor to the size of multi-agent meetings. As the number of potential transaction partners increase, the cost of attending such a meeting increases.

(c) **Joint Ventures**: Joint ventures can occur in either a multilateral meeting or in a bilateral meeting. In a joint venture, Agents come together to create a new Agent. The new Agent receives an injection of Financial Funds, Experience Funds and Assets from the "parent" Agents. The founding Agents continue to exist independently, while receiving a variable flow (proportional to their investment in the joint venture) of Financial Funds from the Agent created as a result of the joint venture.
(d) **Mergers**: Mergers can occur in either a multilateral meeting or in a bilateral meeting. In a merger, Agents come together to create a new Agent, by pooling all their Assets, Financial Funds and Experience Funds. The original Agents cease to exist as independent Agents, and will instead be represented by this new Agent.
(e) **Subsidiaries**: The creation of subsidiaries is a transaction undertaken by an Agent alone. In creating a subsidiary, one Agent unilaterally creates a new Agent, which receives an injection of Financial Funds, Experience Funds and Assets from the "parent" Agent. The original Agent continues to exist independently, and earns a variable flow of Financial Funds dependent on the success of the subsidiary.[16]

In a multi-Agent meeting, after the presentation of Trading Sets, the participants first decide if they wish to form a Merger, or create a Joint Venture. If they choose not to do so, the meeting will break up into a sequence of bilateral meetings. The difference is that Agents in these subsequent bilateral meetings no longer need to pay meeting costs, but are free to transact.

B.4 Background Activity in the Model

1. **Obsolescence Decay**: Obsolescence decay is the loss of value of existing Assets due to obsolescence over time. This is represented as the shrinking of the Revenue Multiplier of existing Assets over time. The rate of decrease is a function of the number of new Assets generated this period.
2. **Diffusion Decay**: Diffusion decay is the loss of value of existing Assets due to the unintended diffusion of these Assets. There are two aspects to this diffusio: diffusion to Agents within the model, and diffusion to parties outside the model. .
   (a) **Diffusion Decay to Agents**: Diffusion of Assets to Agents within the model allows beneficiary Agents to use the Assets competitively. This form of decay is represented as the random distribution of Assets within the model to a greater number of Agents. Each Agent coming into the possession of these new Assets will make independent decisions whether to incorporate it into its Active or Passive sets, or to ignore it (if it is fully laden);

[16] An Agent that finds itself with far too many unrelated Assets may carve off portions of it into subsidiaries.

Table 4. Multi-agent meetings & 2-agent meetings

	Multi-Agent Meeting	**2-Agent Meeting**
Arrangement Cost	Pays arrangement cost for (N-1) Agents that each Agent elects to meet.	If the meeting is arranged, the Agent pays the arrangement cost.
Presentation Cost	Each Agent presents once at each meeting, and pays the presentation cost for the Trading Set it presents once for each meeting it attends.	
Inspection Cost	Each Agent inspects (N-1) Trading Sets, and pays the inspection cost for inspecting the (N-1) Trading Sets.	Each Agent inspects one other Trading Set, and pays the inspection cost for inspecting that one Trading Set.
Fixed Cost	Each Agent pays the Fixed cost of a meeting once for each meeting it attends.	
Trading Set Presented	Based on lowest degree of recurrence with all (N-1) other Agents.	Based on degree of recurrence with the other participating Agent.

(b) **Diffusion Decay to World**: Diffusion of Assets to parties outside the model causes the beneficiaries to reduce their demand for the Asset in the model. This reduced demand is represented as a random decrease in the Revenue Multiplier of the Asset.

The probability and rate at which Assets will diffuse is a function of its Abstraction, and Codification.

B.5 An Example

Structure of the Model For this example we use a 5x5x4 I-Space Matrix, i.e. 5 levels of each Abstraction (A), and Codification (C), and 4 levels of Diffusion (D). In this case, 0 is the minimum Abstraction, Codification, and Diffusion; 3 is the maximum level of Diffusion; and 4 is the maximum level of Abstraction and Codification.

In this model, the corresponding Base Revenue Potential for each location in I-Space is calculated as ((A+1) * (C+1)) / ((D+1) * 25).

Fig. 3 shows the I-Space Matrix, and the associated Base Revenue Potentials for each location in the I-Space. In order to show the I-Space Matrix – which is a 3-dimensional matrix, we will view the I-Space Matrix in 2-dimensional slices, where each slice represents a different level of Diffusion.

Table 5. I-Space matrix: base revenue potential view

Cod	(Diffusion=0)						Cod	(Diffusion=1)					
4	0.20	0.40	0.60	0.80	1.00		4	0.10	0.20	0.30	0.40	0.50	
3	0.16	0.32	0.48	0.64	0.80		3	0.08	0.16	0.24	0.32	0.40	
2	0.12	0.24	0.36	0.48	0.60		2	0.06	0.12	0.18	0.24	0.30	
1	0.08	0.16	0.24	0.32	0.40		1	0.04	0.08	0.12	0.16	0.20	
0	0.04	0.08	0.12	0.16	0.20	Abs	0	0.02	0.04	0.06	0.08	0.10	Abs
	0	1	2	3	4			0	1	2	3	4	
Cod	(Diffusion=2)						Cod	(Diffusion=3)					
4	0.07	0.13	0.20	0.27	0.33		4	0.05	0.10	0.15	0.20	0.25	
3	0.05	0.11	0.16	0.21	0.27		3	0.04	0.08	0.12	0.16	0.20	
2	0.04	0.08	0.12	0.16	0.20		2	0.03	0.06	0.09	0.12	0.15	
1	0.03	0.05	0.08	0.11	0.13		1	0.02	0.04	0.06	0.08	0.10	
0	0.01	0.03	0.04	0.05	0.07	Abs	0	0.01	0.02	0.03	0.04	0.05	Abs
	0	1	2	3	4			0	1	2	3	4	

The I-Space Matrix above represents the underlying structure, showing how revenue is distributed within I-Space.

We now look at the individual Agents participating in this I-space. Each Agent will have its own set of Nodes and Links that reside in various locations in the I-Space matrix. For instance, Agent 1 might have a distribution of Nodes (N_x) and Links (L_y) that looks like Table 6.

Table 6. I-Space matrix: agent asset distribution view

Cod	(Diffusion=0)						Cod	(Diffusion=1)					
4			N_1	N_2	N_3,N_4		4						
3							3		L_2	N_5,L_3			
2							2						
1		L_1					1		N_6		N_7		
0						Abs	0						Abs
	0	1	2	3	4			0	1	2	3	4	
Cod	(Diffusion=2)						Cod	(Diffusion=3)					
4							4				L_7		
3			L_4	L_5			3				L_8	L_9,L_{10}	
2					L_6		2						
1		N_8					1	N_{10}					
0			N_9			Abs	0						Abs
	0	1	2	3	4			0	1	2	3	4	

From this view, we see that Agent 1 has 10 Nodes and 10 Links. The general distribution indicates its Links are mostly Abstract and Codified and Diffused, while its Nodes are mostly Abstract, Codified and unDiffused.

Combining the information from Fig. 3 and Fig. 4, we can thus tell, for instance, that Node #7 is fairly Abstract (3), fairly unCodified (1), fairly unDiffused(1) and will generate a Base Revenue of 0.24 units per unit time. If Node #7 were a basic Node, it would have a Revenue Multiplier of 1 (barring Obsolescence), and would generate revenue of 0.24 units per unit time.

Beyond the I-Space, at any point in time, a global Linkage Probability Matrix exists which maintains the Linkage Probabilities between all existing Node-Link pairs.

From the perspective of an Agent who has limited memory, only a subset is relevant. In this case, where an Agent has an Active Set size of 5 of each type, the relevant subset of the Linkage Probability Matrix might look like Table 7.

Table 7. Linkage probability matrix: agent subset

		Active Node					Passive Node				
		N_2	N_3	N_4	N_8	N_{10}	N_1	N_5	N_6	N_7	N_8
Active Links	L_1	0.22	0.30	-	0.16	0.20	-	-	0.14	-	-
	L_3	-	-	0.23	0.25	0.26	-	-	-	0.15	-
	L_4	0.24	0.23	-	0.17	0.18	-	-	0.14	-	0.17
	L_6	-	-	0.28	0.23	0.18	-	-	0.15	-	0.13
	L_{10}	0.22	0.21	-	0.20	0.28	-	0.13	-	-	-
Passive Links	L_2	0.11	0.14	0.14	-	-	0.13	0.12	-	0.16	0.15
	L_5	-	-	-	0.14	-	-	0.20	0.18	0.16	-
	L_7	-	-	0.23	-	0.13	0.20	-	-	0.16	0.15
	L_8	-	-	-	-	-	-	0.18	0.20	0.16	0.14
	L_9	-	-	-	-	-	-	0.17	0.15	-	0.17

The above indicates that for instance, Link #1 has fair Linkage Probability with Node #3, and the Node #10 has a fairly high Linkage Probability with Link #10. As a whole, we also see that Agent 1 appears to have chosen a set of Nodes and Links that have higher than average mutual Linkage Probabilities and placed them in the Active Set.

The Linkage Probability Matrix also shows implied relationships between Assets of the same type. Node #2 is a direct competitor with Node #3, because for every Link that Node #2 has some non-zero Linkage Probability with, Node #3 has a non-zero Linkage Probability as well.

On the other hand Node #4 is completely unrelated to Node #2 and Node #3, because for every Link that Node #2 and Node #3 have some non-zero Linkage Probability with, Node #4 has Linkage Probability of zero, and vice versa.

B.6 Asset Evolution in the Example

Over time, an Agent's portfolio of Assets may change. If we return to Agent 1 after one time period, its portfolio may have been updated to look like Table 8 (changes in **bold**).

Table 8. I-Space matrix: agent asset distribution view (one period later)

Cod	(Diffusion=0)						Cod	(Diffusion=1)					
4			N_1	N_2	N_3		4					**N_4**	
3							3		L_2	N_5,L_3			
2							2		**N_{11}**				
1	**N_{12}**	L_1					1		**$N_6\uparrow$**		N_7		
0						Abs	0						Abs
	0	1	2	3	4			0	1	2	3	4	
Cod	(Diffusion=2)						Cod	(Diffusion=3)					
4							4				L_7		
3			L_4	**$L_5\rightarrow$**	**L_{11}**		3				L_8	L_9,L_{10}	
2					L_6		2						
1		N_8					1	N_{10}					
0			N_9			Abs	0						Abs
	0	1	2	3	4			0	1	2	3	4	

In this updated view, we see that Node #4 has increased in Diffusion, Node #6 has been codified, and Link #5 has been abstracted. In addition, a new Node #12 has appeared.

The increase in Diffusion for Node #4 could have come about in a variety of ways. It might have been the result of Licensing by Agent #1, or it might have been the impact of Diffusion Decay.

Furthermore we have new Node #11 – the result of codifying Node #6, and a new Link #11 – the result of abstracting Link #5.

Finally we have Node #12 – a new Node that has not been created as a result of any of the "moves" in I-Space. There exists two possibilities for the existence of Node #12, it can either be the result of:

- creation of a new Asset from a set of Nodes and Links that have sufficiently high mutual Linkage Probability; or
- Agent 1 being the beneficiary of some form of Diffusion, whether intended (through Trading) or unintended (through Diffusion Decay).

In this case, if Node #12 were the result of (a), and the threshold Linkage Probability for creating new Nodes was 0.25, we can examine the earlier

Linkage Probability Matrix to see what are the likely constituents of this Node #12.

Revisiting our original Linkage Probability Matrix (Fig. 5) – we find that only the chain of Nodes #8, #10 and Link #3, have Linkage Probabilities that meet this requirement, are thus likely to be the constituents of Node #12.

In this new time period, the Linkage Probability Matrix will also have changed as Agent 1 makes its own decisions to include/exclude new Assets. A possibility might look like Table 9 below (changes in **bold**).

Table 9. Linkage probability matrix: agent 1 subset (one period later)

		Active Node					Passive Node				
		N_2	N_3	N_4	N_8	N_{10}	N_1	N_5	$\mathbf{N_{11}}$	N_7	N_8
Active Links	L_1	0.22	0.30	-	0.16	0.20	-	-	**0.17**	-	-
	L_3	-	-	0.23	0.25	0.26	-	-	**-**	0.15	-
	L_4	0.24	0.23	-	0.17	0.18	-	-	**0.14**	-	0.17
	L_6	-	-	0.28	0.23	0.18	-	-	**0.20**	-	0.13
	L_{10}	0.22	0.21	-	0.20	0.28	-	0.13	**-**	-	-
Passive Links	L_2	0.11	0.14	0.14	-	-	0.13	0.12	**-**	0.16	0.15
	L_{11}	**-**	**0.15**	**-**	**0.14**	**-**	**0.16**	**0.20**	**0.18**	**0.16**	**-**
	L_7	-	-	0.23	-	0.13	0.20	-	**-**	0.16	0.15
	L_8	-	-	-	-	-	-	0.18	**0.23**	0.16	0.14
	L_9	-	-	-	-	-	-	0.17	**-**	0.15	0.17

Among the Nodes, Node #6 was codified to create Node #11. Therefore, by definition, Node #11's Linkage Probabilities are derived from Node #6's, with higher non-zero Linkage Probabilities as a result of codification. As Node #11 is superior to Node #6, the Agent has chosen the former to replace the latter in its Passive set.

Similarly, among the set of Links, Link #5 was abstracted to create Link #11, and for similar reasons, the latter has replaced the former in the Agent's Passive set. As defined, Link #11's Linkage Probabilities are essentially those of Link #5, with more non-zero Linkage Probabilities as a result of abstraction.

Finally, aside from the changes listed above, other changes will be taking place in the background. Among others, Obsolescence Decay will have set in, and the various Nodes and Links that existed will now generate less revenue

per unit time. Diffusion Decay will also have occurred, and we saw a possible effect of Diffusion Decay on Node #4 earlier (although this increase in Diffusion might have also been the result of sharing).

B.7 Agent Interaction in the Example

Stepping back from the Agent 1, we look at the I-Space world that Agent 1 resides in. In this example, there are seven other Agents in Agent 1's world (Agent 1, Agent 2... Agent 8).

If this is the very first period in the simulation, Agent 1 will not have met any other Agent. However, if we visit Agent 1 some time into the simulation, Agent 1 will have developed a history of interactions with other Agents, and will have established a set of preferences as to which Agent it would like to interact with. Similarly for other Agents, each of them will have their own list of Agents they would prefer to interact with.

At the beginning of a period, each Agent will update its disposition to other Agents. A possible scenario is shown below, each row is an Agent's disposition to the Agents in the columns, i.e. the entry in row i, and column j, indicates Agent i's disposition towards Agent j – where "2" indicates a strong desire for meeting; "1" amenable to meeting; and "0" no desire for a meeting at all.

Table 10. Agent meetings: arranging meetings

	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8
Agent 1		1	2	0	1	1	1	1
Agent 2	1		0	1	1	2	1	2
Agent 3	1	0		0	1	1	1	0
Agent 4	2	1	1		0	0	1	2
Agent 5	1	1	1	0		0	0	1
Agent 6	1	0	1	1	1		1	1
Agent 7	0	0	2	0	1	1		2
Agent 8	0	0	0	1	2	0	0	

In this example, Agent 1 does not wish to meet Agent 4, but is amenable to a meeting if arranged b Agents 2, 5, 6, 7, and 8. In addition, it has a strong desire to meet Agent 3, and will thus attempt to arrange a meeting with Agent 3.

Because Agent 3 is amenable to a meeting with Agent 1, it will respond favorably to Agent 1's invitation.

On the other hand, Agent 2 will similarly try to engage Agent 6 in a meeting, but as Agent 6 has no desire to meet Agent 2, no meeting will occur between Agent 2 and Agent 6.

In addition to arranged meetings, random encounters may give rise to meetings as well. The figure below shows the meetings of Agents, after taking into account Agent disposition that leads to meetings being arranged, as well as random encounters that lead to Agents meeting other Agents. A "4" in row i, and column j, indicates that Agent i will meet Agent j in a meeting. At this point, the type and size of meeting is undetermined because Agents will aggregate as many as possible meetings with other Agents to reap economies of scale in multi-Agent meeting.

Table 11. Agent meetings: actual meetings

	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8
Agent 1		•	•	•		•		•
Agent 2								
Agent 3					•		•	
Agent 4						•		•
Agent 5							•	•
Agent 6								•
Agent 7								•
Agent 8								

Based on the matrix, a 4-Agent meeting is possible between Agent 1, Agent 4, Agent 6 and Agent 8.

At the same time, a 3-Agent meeting is possible for Agent 3, Agent 5 and Agent 7. Or for Agent 5, Agent 7 and Agent 8.

Because Agents want to attend the largest possible meeting, Agent 8 will eschew the 3-Agent meeting in favor of the 4-Agent meeting.

Looking at the 4-Agent meeting between Agent 1, Agent 4, Agent 6 and Agent 8. Of the four Agents participating, the meeting between Agent 4 and Agent 8 was arranged by Agent 4 – thus Agent 4 has incurred Arrangement cost for this meeting. When the four agents do meet, they pay the Fixed cost of the meeting. All this is before the Agents even have had the chance to present their Trading Sets or inspect the other Agents' Trading Sets.

Finally, when the Agents meet, each Agent will make available its Trading Set. For Agent 1, it has a total of 10 Nodes and 10 Links available. In our model, with Abstraction/Codification values of 0∼4, Trading Sets are defined as follows:

In our example, Agent 1 has met Agent 4 six times before, Agent 6 and Agent 8 nine times before. Now if Agent 1 had met Agent 4, Agent 6 and Agent 8 separately, it would present Trading Set 2 to Agent 4, and Trading Set 3 to Agent 6 and Agent 8 respectively. However in the context of a multi-party interaction however, it will present the Trading Set based on the Agent with which it has had the least history – in this case Agent 4.

Table 12. Agent meetings: trading sets

Trading Set	**# of Previous Meetings**	**Characteristics of Assets in Trading Set**	**Asset in Agent 1's Trading Set**
0	$0 \sim 1$	Abstraction & Codification =4	N_3, N_4
1	$2 \sim 4$	Both Abstraction & Codification >=3	All above + N_2, L_7, L_8, L_9, L_{10}, L_{11}
2	$4 \sim 7$	Both Abstraction & Codification >=2	All above + N_1, N_5, L_3, L_4, L_6
3	$8 \sim 12$	Both Abstraction & Codification >=1	All above + N_7, N_8, N_{11},L_1, L_2
4	$13 \sim 18$	All Assets	All above + N_9, N_{10}

Thus at the Meeting Agent 1 will only make available Trading Set 2 for inspection. Agent 4, Agent 6 and Agent 8 will also present their Trading Sets. Additional costs are then incurred - Agent 1 incurs a Presentation cost based on presenting Trading Set 2, and it incurs an Inspection cost based on inspecting the Trading Sets of Agent 4, Agent 6 and Agent 8.

After Agents have inspected the Trading Sets of all participants, the meeting has the opportunity to decide if it wishes to combine in a Merger, or create a Joint Venture. If they choose not to do so, the Agents are then free to engage in bilateral transactions with the participants. The bonus of the multi-Agent meeting is that because they have all paid their presentation and inspection costs, they will be able to engage in three separate bilateral meetings, having only paid the presentation cost once.

Note that there is no requirement for Agent 1 to collaborate with either Agent 4, Agent 6 or Agent 8, it can do so, with all of them, some of them or none of them. The multi-Agent meeting only serves to amortize to cost of presentation over a larger audience, and really only becomes effective when all parties have a similar history of collaboration/meetings (otherwise the Agent can only present the most Abstract and Codified Assets and is unable to collaborate with its more concrete and uncodified Assets).

References

[Boi98] Max Boisot, *Knowledge assets: Securing competitive advantage in the information economy*, Oxford University Press, Oxford, UK, 1998.

[Daw82] Richard Dawkins, *The extended phenotype : The gene as the unit of selection*, Oxford University Press, Oxford, UK, 1982.

[Daw89] ———, *The selfish gene*, Oxford University Press, New York, NY, 1989.

Part III

Communication

On Representing Special Languages with FLBC: Message Markers and Reference Fixing in SeaSpeak

Steven O. Kimbrough[1] and Yinghui (Catherine) Yang[2]

[1] University of Pennsylvania, Philadelphia, PA 19104, USA,
kimbrough@wharton.upenn.edu
[2] University of California at Davis, Davis, CA, USA,
yiyang@ucdavis.edu

Abstract. SeaSpeak is "English for maritime communications." It is a restricted, specially-designed dialect of English used in merchant shipping and accepted as an international standard. This paper discusses, in the context of SeaSpeak, two key problems in the formalization of any such restricted, specially-designed language, viz., representing the illocutionary force structure of the messages, and formalization of such reference-fixing devices from ordinary language as pointing and use of demonstratives. The paper conducts the analysis in terms of Kimbrough's FLBC agent communication language.

1 Introduction

SeaSpeak is known as "English for maritime communications." It is the language of merchant shipping, a restricted, artificial, specially-developed, English-like language adopted in 1988 by the International Maritime Organization (IMO) of the United Nations for use in ship-to-ship and ship-to-shore communications. Part of the significance of SeaSpeak's success is that it demonstrates the value and use of specially-built artificial languages. The question then naturally arises of whether a designed special language might be fully formalized and used in machine-to-machine or human-to-machine communication. We have been intrigued by such possibilities and in consequence have been investigating SeaSpeak to this end.[1] In what follows we focus on two aspects of the larger programme of formalizing special languages:

- Illocutionary forces
- Reference fixing

These aspects of language, discussed in detail herein, are quite common. They occur in SeaSpeak, but also in nearly any special language that will be interesting. Our chosen vehicle of formalization, Kimbrough's FLBC, is also a

[1] [KLPY03,KY04]

special case. SeaSpeak and its ilk present an important test challenge for *any* agent communications language (ACL, of which FLBC is an instance). We shall present evidence in the form of analysis that indeed FLBC is adequate to the problems of representing illocutionary forces and fixing reference in SeaSpeak. The exercise and the lessons learned will apply in general to ACLs.

That is the overview. Details begin in the next section with some background on special languages.

2 Special Languages

Language enables communication. Languages inhibit it, for communication requires a common language and the cost of learning multiple languages raises an often unsurmounted barrier. Having a *lingua franca*, a general language known universally, would afford universal communication. At various times and places certain natural languages, such as Greek, Latin, Mandarin, and French, have approximated universal communication vehicles.

Today English in some form appears headed towards being the universal language of commerce and affairs. The fact remains, however, that universal proficiency in English is not around the corner. Further, even with universal fluency in English there are, and will always be, realms of discourse for which precise and accurate communication is required concerning specialized topics. It is not enough to have basic knowledge of English if the purpose of communication is air traffic control, navigation, law enforcement, and so on. In these and many other realms of discourse there exist specialized concepts and vocabulary that have to be mastered in the interests of efficient and effective communication. General fluency in English is not sufficient. Neither is it necessary.

Special languages can in principle be created that are relatively easy to learn and that are sufficiently expressive for particular purposes. They need be mastered only by a given community of interest. This idea has had an extensive history and considerable uptake, and it goes by a number of names. Including *planned languages*, the literature uses a number of other terms and recognizes a number of related concepts:[2]

> artificial languages, constructed languages (conlangs), invented languages, imaginary languages, fictional languages, etc., including universal languages, auxiliary languages, interlanguages or interlinguas, international languages; and also including logical languages, number languages, symbolic languages, etc. [Har02]

as well as others, including *restricted languages*, *designed languages*, and *sublanguages*.

[2] Many of these terms denote different, albeit related, concepts. We shall use *special language* as an umbrella term.

> Informally, we can define a sublanguage as the language used by a particular community of speakers, say, those concerned with a particular subject matter or those engaged in a specialized occupation.[Sag86, page 2]

Several artificial sublanguages have been fielded, and are successfully in use today.[3] Examples include AirSpeak, SeaSpeak, PoliceSpeak, and LinguaNet.[4] These languages were designed to be easily learned so that they can be spoken and heard effectively. Their employment and continued development today suggests they will be useful in the longer term. They employ a controlled, or restricted, vocabulary. It is typically true of sublanguages that their

> ... grammar contains additional rules not satisfied by the language as a whole. It also happens that some of the grammatical rules of the language as a whole disappear, i.e., do not apply, in a sublanguage. Since the sublanguage must satisfy the rules for the language, this disappearance is possible only if the rules are satisfied vacuously in the sublanguage, i.e., if certain word classes or well-formed sequences or transformations do not appear in the sublanguage. [Har68, page 154]

Telegraphic languages, yet more austere forms of sublanguage, are also widely in use and readily display the simplified, constrained grammar characteristic of sublanguages. We note that *telegraphic language* and *telegraphic speech* are also terms of art in the field of child development, and it is here that the terms obtained their original meaning.

> When children initially produce grammar, their language often sounds rather like the abbreviated language of telegrams ("Daddy gone," "Mummy shoe." "See big car"). This is why, in the past, this type of early output was referred to as telegraphic speech. At this stage, toddlers omit indefinite and definite articles, as well as prepositions and the like. They also leave out morphemes like plural "s," progressive "ing," and possessive "'s." [KK01, page 94]

Telegraphic languages are not unknown among adults. Fitzpatrick et al. [FBH86] present a particularly clear case study of a telegraphic language used in the U.S. Navy. The stylization apparent in examples from this language—e.g., "72 manhours expended," "Stock requisition shipped," "Work request submitted," "Improper repair work performed," "No parts required" [FBH86, page 45]—will be familiar to the reader.[5]

[3] We shall not discuss various more ambitious efforts to develop general-purpose universal languages. Esperanto is perhaps the most well-known candidate language. For relevant background see [Lar85,Mac30,Ogd38,Ric43,Swa80]; also, Harrison [Har02] has put together a very useful bibliography.

[4] See [Ben03], [Joh98], [Joh02], [Pro03], [Lin03], and [J$^+$93] for an overview.

[5] Portions of this section contain a revised version of material appearing in [KLPY03].

3 Two Problems

Can special languages—especially artificial languages, sublanguages, and telegraphic languages—be formalized so that machines may productively conduct inferences using them? The question is significant both theoretically and and for applications, as has been noticed, e.g.,

> An interesting suggestion which could have widespread applications is that particular subdisciplines do in practice use a limited set of grammatical structures as well as a restriced vocabulary: a sublanguage or metalanguage, easily comprehended by those within the subdiscipline but foreign to the layman. Because of the limited number and specialized nature of the grammatical structures found it becomes possible to apply content analysis techniques to texts within the subdiscipline with success consistently. The particular subdiscipline illustrated is pharmacology, but it seems likely that the approach would be valid in any of the 'hard' sciences with a clearly defined vocabulary (jargon!) and generalized methodology. [Fos82, pages 51–2]

(See also [Sag75].) Theoretically, the question presents an apt challenge for ACLs (agent communication languages), including the various projects to create FLBCs (formal languages for business communication), and the various XML representation efforts. Can the ACLs adequately represent a given artificial language? If not, how might they be improved? What does formalization of artificial languages tell us about requirements for ACLs? From a practical point of view, formalization could afford human-machine and machine-human communication, including language translation and error detection, as well as machine-machine communication, with its attendant possibilities of reducing time and labor costs. Perhaps of most immediate use, formalization and structuring present opportunities for automated recovery and discovery of information.

These are large and fascinating questions, which succinctly put the context for the results reported in this paper. We essay here to make a modest, yet discernible, contribution to the advance on them. We shall examine one (informal) artificial language—SeaSpeak—and one variety of ACL (agent communication language), Kimbrough's FLBC, based on event semantics, thematic roles, and disquotation of propositional content. (For background on FLBC see in this volume "A Note on Modeling Speech Acts as Signalling Conventions" [JK04] and "Practical Contract Storage, Checking, and Enforcement for Business Process Automation" [AEB04].[6]

Specifically, SeaSpeak is a notable example of an artificial sublanguage, which is established and used successfully, and which might benefit from formalization. We have been investigating the prospects for such a formalization

[6] Other references for FLBC include [Kim90], [Kim99], [KM97], [KT00], [Kim01], [Kim02], [KLPY03], and [KY04].

(see [KLPY03,KY04]), with positive results, both for SeaSpeak in particular and for artificial sublanguages more generally. In what follows, we address in detail two key technical issues whose resolution is essential for any programme of formalizing communication among artificial (and human) agents in other than very restricted domains. The two issues are:

- The speech act structure of SeaSpeak messages.
- Dynamic reference fixing in SeaSpeak messages.

Central to speech act theory[7] is the distinction between the *illocutionary force* and the *propositional content* of an utterance. This distinction, commonly thought to originate with Austin [Aus62], goes back at least to Charles Sanders Peirce, in the nineteenth century.

> Like other philosophers of thought and language, Peirce distinguished the force of an utterance from its propositional content. A proposition can be 'affirmed, denied, judged, doubted, inwardly inquired into, put as a question, wished...' *Assertion* of a proposition involves 'the deliberate exercise, in uttering a proposition, of a force tending to determine a belief in it in the mind of the interpreter' .... Assertion involves 'taking responsibility' for the truth of the proposition. [Hoo02, page 62]

In any event, under the perspective of speech act theory, which is widely accepted and which we accept,[8] every utterance (in any language) may be analyzed as having an $F(P)$ structure: an illocutionary force, F, is applied to a propositional content, P. In general, illocutionary forces and propositional contents are in a many-to-many relationship. One force—e.g., asserting, directing, promising—may be applied to many propositional contents—e.g., 'I will arrive tomorrow', 'The tide will come in at 6 p.m.', 'An act of nuclear terror will strike New York City within 10 years'. Similarly, one propositional content may be the object of several different illocutionary forces. One may assert it, deny it, promise it, and so on. If this most basic tenet of speech act theory is correct, then it should be possible to recognize speech acts in SeaSpeak and to say something in general about their structures. In fact, as we shall see, SeaSpeak presents a happy confirmation of the $F(P)$ thesis and, as we shall argue, the SeaSpeak illocutionary forces (called *message markers* in SeaSpeak) may be aptly represented in FLBC.

The problem of dynamic reference fixing arises outside the perspective of speech act theory. We present and discuss the problem in depth in §6. Briefly, it is this. In communicating we wish to talk about things, about

[7] Classically, [Aus62], [Lev83], [Sea69], and [SV85]; [BH79] is useful; [LAP04], and [KM97] present application-oriented summaries.

[8] The Language/Action Perspective community has been holding workshops and producing papers for some years now, promoting and articulating applications of speech act theory. See [LAP04].

shoes, ships, sealing wax, cabbages, kings, kinghood, numbers, beliefs, and even things that do not exist such as unicorns. To talk about any thing we need to make reference to it. If the thing has a proper name, this is relatively unproblematic. Usually, however, we make reference to something with a *the*-expression—as in *The cat is on the mat*—or some similar device. The problem of dynamic reference fixing for an ACL is the problem of formalizing such expressions as *The cat is on the mat* in such a way that reference is successful and recoverable by the addressee of the utterance. SeaSpeak is replete with such expressions, particularly *the*-expressions, and so provides an opportune context in which to tackle this problem.

All of this requires a bit of background on SeaSpeak. To that now.

4 Background on SeaSpeak

SeaSpeak was developed and deployed in consequence of vastly increased shipping during the 1960s and 1970s. At the same time, the distribution of nationalities of ships' officers gradually changed from roughly 80% English-speaking and 20% other to roughly 80% other and 20% English-speaking. The need for regularization of practices in one language and the training of officers in its use was therefore agreed, and English, already the language of civil aviation, was chosen by the IMO.

During 1982–1983, SeaSpeak was created by specialists in maritime communications and applied linguistics [Joh02]. SeaSpeak is a system for speech communication, and it is intended for use in situations where it is essential that communication should be as clear, brief and accurate as possible.

Like SeaSpeak, Airspeak and Policespeak are also special purpose systems for speech communication among targeted users. They are the special languages of command and control where the utterances you make affect something far away—applying to communication between ships, for air traffic control and in police operations. Edward Johnson, Senior Fellow of Wolfson College, University of Cambridge, U.K., has been a pioneer in the field of operational and communication languages. Johnson was responsible for formulating an international language for maritime communication, SeaSpeak (1982), an air traffic pilot training communication program, AirSpeak (1986), and a restricted operational language and set of procedures for police communication, PoliceSpeak (1987).

SeaSpeak regulates ways of speaking and ways of establishing a conversation. It defines a technical vocabulary. All messages begin with a *message marker* that indicates the nature of what follows, such as advice, information, instruction, intention, question, request, warning, or a response to one of these. The definitive reference for SeaSpeak is the *SeaSpeak Training Manual* [WGJS88], upon which we draw for our analysis and formalization.

From a formalization perspective the central concept in SeaSpeak is the message marker. The manual has this (and not much else) to say about message markers in general.

> Maritime messages transmitted over VHF should be short, accurate, and relevant. Furthermore, messages should be transmitted in language simple enough for a non-native speaker of English to comprehend without difficulty.
> One useful means of making the language simpler is to indicate, at the beginning of a message, what sort of message it is going to be. Thus, if a question is going to be asked, the speaker simply says the word 'QUESTION' before the question itself. Similarly, if a piece of advice is going to be given, the speaker says the word 'ADVICE' in advance of his message. There are just severn of these *Message Markers* and after a little practice, learners should experience no difficulty in using them.
> These *Message Markers* have another function: that of imposing order on the conversation, since each message marked in this way requires a reply correspondingly marked (even if that reply is nothing more than an acknowledgement of the message received). This procedure helps to ensure that:
> 1. messages do not become confused with each other
> 2. each message is dealt with in turn
> 3. a participant receiving a reply knows which message is being replied to.
>
> [WGJS88, page 96]

SeaSpeak has only seven markers (with a mirroring reply-marker in each case). The seven are [WGJS88, pages 96–7]:

1. Information (Information-Received)
2. Warning (Warning-Received)
3. Intention (Intention-Received)
4. Request (Request-Received)
5. Advice (Advice-Received)
6. Instruction (Instruction-Received)
7. Question (Answer)

SeaSpeak's message markers are, as we shall analyze them, essentially speech act operators or illocutionary force indicators. SeaSpeak sentences have, we shall argue, the $F(P)$ structure posited by speech act theory. The Fs of SeaSpeak are its message markers. The content that they govern is simple in form, although not entirely specified and closed. Here is a summary from the training manual.

1. SEASPEAK messages are formed entirely from words within the English language.
2. The total vocabulary used in SEASPEAK comprises 3 kinds of words and expressions:
   (a) **The vocabulary of 'general' English.** Knowledge of the non-specialized vocabulary of English is assumed, and so it is not listed in the SEASPEAK Vocabulary.
   (b) **Words in general maritime use.** These words occur frequently in maritime communications, and are listed in Section I, as Categorised General Maritime Vocabulary.
   (c) **Words in specialised maritime use.** These words and expressions may occur only rarely in general maritime use, but frequently in particular circumstances or for specific communication subjects. They are listed in Section II under the Major Communications Subjects.

[WGJS88, page 160]

Item (2a) presents a particular challenge to formalization efforts. Just what is the scope of 'general English'? Examples are useful. Here and in the sequel we will use *italic font* for messages in SeaSpeak. The following examples are from [WGJS88, page 97].

1. *QUESTION: What is your ETA at the dock entrance?*
2. *INSTRUCTION: Go to berth number: two-five.*
3. *ADVICE: Anchor, position: bearing: one-nine-four degrees true, from Keel Point distance: one mile.*
4. *REQUEST: Please send, quantity: five acetylene cylinders.*
5. *INFORMATION: The pilot is waiting now, position: near buoy number: two-six.*
6. *WARNING: Buoy number: two-five and buoy number two-six are unlit.*
7. *INTENTION: I intend to reduce speed, new speed: six knots.*

We cannot fully address here the use of 'general English' in SeaSpeak. Instead, we focus on formal analysis of the seven message markers and on dynamic reference fixing, particularly use of *the*-expressions, as seen in the examples above.[9]

5 Prototype Example: INFORMATION

Our purpose in this section is to present a prototype, or illustrative, example of representing a SeaSpeak sentence in FLBC. Consider then the simple SeaSpeak sentence:

[9] Portions of this section contain a revised version of material appearing in [KLPY03].

SeaSpeak Sentence 1 *INFORMATION: No vessels are at the anchorage.*

We'll begin by analyzing the content of the simple sentence, *No vessels are at the anchorage,* deferring briefly discussion of the illocutionary force indicator, *INFORMATION.* Thus,

SeaSpeak Sentence 2 *No vessels are at the anchorage.*

Note that *the anchorage* is a referring expression whose meaning cannot be recovered from the bare sentence. To begin, we assume for the sake of the example that a definite anchorage has been identified, called (having proper name) *XBar-Harbor-B.* Later we will discuss at length our analysis of and approach to such expressions as *the anchorage* and *the harbor.*

The first step in formalizing a given SeaSpeak sentence is to convert it to more transparent forms, while remaining in natural (informal) language. The alternative forms are called *stylistic variants.* They are intended to be adequately similar in meaning (for the purposes at hand) to the original sentence, while also being more transparent for purposes of formalization. In this example we eliminate the referring *the*-expression in favor of its corresponding proper name. The first stylistic variant is thus:

SeaSpeak Sentence 3 *No vessels are located at XBar-Harbor-B.*

Notice that sentence 3 is arguably a clearer version of sentence 2, since the answer to the question Which anchorage? may be read off by inspection. We need one further transformation. The result is rather stilted, but fits the perspective of event semantics as deployed by FLBC.

SeaSpeak Sentence 4 *(1) There is a state of being a vessel, s_1, at this time. (2) Nothing now in the vessel state, s_1, is located at XBar-Harbor-B.*

SeaSpeak Sentence 4 is a well-structured stylistic variant of a SeaSpeak *sentence*; it does not constitute a complete SeaSpeak *message*, if only because the speaker and addressee are not identified. Figure 1 contains a complete representation (with comments) of a SeaSpeak *message.* Note that it is divided into three parts:

1. Reference-fixing
2. Presupposition
3. Message body

Points arising:

1. Fully articulated the message body is

   SeaSpeak FLBC Sentence 2 $information(e_1) \wedge Speaker(e_1, s) \wedge Addressee(e_1, a) \wedge Cul(e_1, now) \wedge Object(e_1, \lceil \forall x(In(x, s_1) \rightarrow \neg Located(x, a_1)) \rceil)$

1. Reference-fixing
   s = the speaker
   a = the addressee
   a_1 = *XBar-Harbor-B* (the name of a particular anchorage)
2. Presupposition
   (a) There is a state, s_1, of being a vessel at this time:
   $vessel(s_1) \wedge\ Hold(s_1,\ now)$
3. Message body
   (a) Propositional content.
   $\forall x(In(x, s_1) \rightarrow \neg Located(x, a_1))$
   (b) Illocutionary force.
   Further, the utterance is in the class INFORMATION, which indicates its illocutionary force. SeaSpeak FLBC Sentence 1 represents the INFORMATION force for the utterance to hand, with ϕ serving as a place-holder for the propositional content.
   SeaSpeak FLBC Sentence 1 $information(e_1) \wedge\ Speaker(e_1, s) \wedge\ Addressee(e_1, a) \wedge\ Cul(e_1,\ now) \wedge\ Object(e_1, \lceil\phi\rceil)$

Fig. 1. Basic example for FLBC representation of SeaSpeak INFORMATION messages

2. Rendered back into still stilted English, the message body says that e_1 *is an INFORMATION utterance event, occurring now, whose speaker is s, whose addressee is a, and whose propositional content is that no vessel is located at* a_1.
3. SeaSpeak FLBC Sentence 2 is a formalized representation of SeaSpeak Sentence 1 and its stylistic variants. None of these sentences is by itself a complete message. Abstracting Figure 1 to a more general template, we will use the message body item (#3) for representing SeaSpeak sentences.
4. The reference-fixing item (#1) of Figure 1 identifies the speaker, the addressee, and any other proper nouns (here, *XBar-Harbor-B*) required for the message. Strictly speaking, this section could be eliminated and the logical names, e.g., s, could be replaced throughout with their associated proper names, e.g., *Land's End Radio*. It is convenient, however, to retain this section.
5. Use of the presupposition section will be discussed in the sequel. Here, it is used to state the presupposition that there is a vessel state, in which something might potentially be in. Again, this is not strictly speaking necessary. One might drop this presupposition in favor of an expanded (but not equivalent) SeaSpeak FLBC Sentence:

   SeaSpeak FLBC Sentence 3 $information(e_1) \wedge\ Speaker(e_1, s) \wedge\ Addressee(e_1, a) \wedge\ Cul(e_1,\ now) \wedge\ Object(e_1,\ \lceil\forall x \forall s_1(vessel(s_1) \wedge\ Hold(s_1,\ now)\ \wedge\ In(x, s_1) \rightarrow \neg Located(x, a_1))\rceil)$

   (Note that now s_1 is being used as a variable, not a constant.)

In sum, the pattern in evidence here represents a SeaSpeak *message* as a structure, consisting of a reference-fixing section, a presupposition section, and a message body section, the latter being used to hold representations of particular SeaSpeak *sentences*. We devote the balance of this paper to articulating this pattern with further examples and in response to certain problems. The bulk of the problems we address fall into the category of how reference (to external objects) may be fixed in an automated or semi-automated system. We finessed that issue for the sake of the example in the present section. We now address it directly.

6 Problems of Reference Fixing

Ordinary discourse, and certainly SeaSpeak discourse, is replete with references to entities of various sorts, including particular objects, places, times, sounds, happenings, lengths, performances, fictional characters, geographical objects, and social entities (cf., [Jac02, pages 300–3]). Consider a simple example from ordinary language:

- *The cat is on the mat*

Both *the cat* and *the mat* are referring expressions. Which cat? Which mat? These references have to be adequately fixed if the hearer is to understand the sentence. In human–human discourse we have a number of devices we employ, often automatically and without deliberation or even awareness, so that both speaker and hearer will know what the sentence is about. Thus, e.g., *the cat* might harken back to an earlier part of the conversation, or a cat might be pointed to (in some way or another) by the speaker at the time of sentence utterance in such a manner that the speaker has good reason to believe that the hearer has "gotten the point." Similarly, *the mat* might refer by conventions of discourse to a particular mat identified earlier in the conversation.

The subject of how reference is fixed in ordinary discourse is a large and fascinating one. Our concern here is the more limited one of how this might be achieved in machine–to–machine (or process–to–process) discourse using a formal language. How might one process point to something in the world and have the other process be able to discern the object of the reference?[10] We turn now to *the*-expressions, which shall constitute the focus of our attention on these matters.

The word *the* is not always unproblematic, especially in the present context. Consider these typical SeaSpeak sentences:

[10] Note we say *discern* rather than *understand*. Computer processes supporting communication need not 'understand' messages, whatever that murky term may mean. If, however, a message is sent referring to an object, the receiving process will need to pick it out, distinguish it, discern it.

SeaSpeak Sentence 5 *QUESTION; What is your ETA at the dock entrance? [WGJS88, page 97]*

SeaSpeak Sentence 6 *INFORMATION: No vessels are at the anchorage. [WGJS88, page 101]*

SeaSpeak Sentence 7 *INFORMATION: The icebreaker intends to assemble the convoy at time: zero-five-three-zero GMT. [WGJS88, page 101]*

SeaSpeak Sentence 8 *INFORMATION: The casualty is approximately, position: North distance: three miles from you. [WGJS88, page 101]*

The first uses *the dock entrance* to refer to a particular dock entrance. Which one? The second sentence uses *the anchorage* in referring to a specific anchorage. Again, which one? In order for communication to be entirely successful the addressee must be able to ascertain, or pick out, the particular dock entrance or anchorage referenced by the speaker. How does this happen? How could this be brought about in an automated (or partially automated) context, as is under discussion here?

If the referents—the dock entrance and the anchorage—are given proper names—say Dock Entrance A and Anchorage D—and these are known and agreed upon prior to communication, then the problem is greatly simplified. Instead of speaking as above, SeaSpeakers could talk as follows.

SeaSpeak Sentence 9 *QUESTION: What is your ETA at Dock Entrance A?*

SeaSpeak Sentence 10 *INFORMATION: No vessels are at Anchorage D.*

This is exactly what we did in §5, naming *the anchorage* with *XBar-Harbor-B*. More generally, it is certainly possible to construct antecedently a table of proper names and their meanings, and to include this table in the working definition of a communications language, whether it be informal, as in SeaSpeak, or formal as in an FLBC. In the case of Maritime communications, listing all ships, harbors, lighthouses, and officers in this way would no doubt be useful. Practically, however, the approach has its limitations. Harbors and lighthouses may be more or less permanent, but ships and officers come and go. Ships may be renamed. Officers may fall ill during a voyage and new officers be declared. How will the list be maintained, how will the language be updated, and how will the updates be promulgated? Making matters worse, many of the things that need to be discussed are ephemeral: storms and other weather patterns, accidents and other incidents at sea, and so on. It is not a practicable possibility to name antecedently such referents as *the storm* in

SeaSpeak Sentence 11 *QUESTION: When will the storm arrive?*

In natural language—spoken and written, including sublanguages such as SeaSpeak—the complications of reference fixing are manifold. This creates enormous difficulties for information retrieval, as is well known and as is so charmingly described in the following passage from a famous empirical study.

> Sometimes we followed a trail of linguistic creativity through the database. In searching for documents discussing "trap correction" (one of the key phrases), we discovered that relevant, unretrieved documents had discussed the same issue but referred to it as the "wire warp." Continuing our search, we found that in still other documents trap correction was referred to in a third and novel way: the "shunt correction system." Finally, we discovered the inventor of this system was a man named "Coxwell" which directed us to some documents he had authored, only he referred to the system as the "Roman circle method." Using the Roman circle method in a query directed us to still more relevant but unretrieved documents, but this was not the end either. Further searching revealed that the system had been tested in another city, and all documents germane to those tests referred to the sysem as the "air truck." At this point the search ended, having consumed over an entire 40-hour week of on-line searching, but there is no reason to believe that we had reached the end of the trail; we simply ran out of time. [BM85]

Our problem is akin to, yet distinct from, the information retrieval problem. The latter aims at finding multiple references to a common object. The question before us is how a speaker may establish reference to an object in such a way that the addressee will be able to distinguish, pick out, establish her own reference to the object.[11]

In seeking to formalize and automate it will be helpful to examine how reference is fixed in natural language, for the devices employed there, or something like them, may prove useful in our context. Linguists and philosophers have discerned a number of reference-fixing devices in natural language, in addition to the use of proper names, already noted. For present purposes, these devices may be divided into *descriptions* and *indexicals*. Reference by description occurs when the speaker is able to describe an object with precision adequate for the purposes to hand, without resorting to a proper name (of the object at the time of the utterance). The notorious example of Prince, a rock musician who unnamed himself and took on a symbol (different name?), is illustrative. Speakers quite successfully referred to him by (definite) description: *the artist formerly known as Prince*. Accepting for the moment

[11] Objects should be understood in a broad sense, to include events, processes, actions, social conventions, and so on, as well as physical objects such as shoes, ships, and sealing wax. Also, we do not say or require that the addressee 'understand' what the speaker is talking about, whatever that may mean. The question is whether the addressee can, operationally, identify the referent. See the article by Jones and Kimbrough in this volume for a discussion of signalling [JK04].

Russell's analysis [Rus05], "The artist formerly known as Prince is giving a concert in Philadelphia tomorrow" will be true just in case:

1. There exists an artist formerly known as Prince,
2. There is only one such artist, and
3. That artist is giving a concert in Philadelphia tomorrow.

(Further complicating the matter, I am told that as of the date of this writing, 29 August 2004, the fellow in question has reverted to calling himself Prince. Is our sentence still true? Well, suppose that Bobby Short, who has never changed his name, is playing at the Café Carlisle tonight. Is it true to say "The artist formerly known as Bobby Short is playing at the Café Carlisle tonight"?)

Besides describing things, the other main reference-fixing device, and perhaps a more basic one,[12] is simply to point to them, to indicate them in one way or another. This might be done non-verbally, say by pointing, nodding, or turning one's head. When it is done verbally, with language, we say that *indexicals* are used. Among them we will count proper names and pronouns. In addition, the *demonstratives*—in English, *this, that, these,* and *those*—are widely used in natural language. Here is an example from SeaSpeak, used for identifying the speaker over a public radio channel.

SeaSpeak Sentence 12 *This is Land's End Radio. [WGJS88, page 39]*

Interestingly, however, the SeaSpeak manual [WGJS88] does not have other *kinds* of examples using demonstratives. (The above form is used throughout to identify speakers.)

The use of *the*, as in SeaSpeak Sentences 5–8, remains unresolved. What are we to make of it? How are we to analyze it, given the linguistic and logical concepts described above?

Note that these SeaSpeak Sentences contain locutions that *resemble* definite descriptions:[13] *the dock entrance* (but there are many docks and each as at least one entrance), *the anchorage* (but there are many anchorages), *the icebreaker* (but there are many icebreakers), *the convoy*, and *the casualty.* Our suggestion is that reference gets fixed antecedently to these locutions, which may then be interpreted as definite descriptions *on the presupposition that the necessary reference-fixing has been done.* One might call this a *distributed definite description.* The suggestion is illustrated, and we think confirmed, by an example from the SeaSpeak manual [WGJS88, page 154]. See Figure 2.

[12] Since Peirce there has been a tradition of viewing demonstratives as the most basic form of reference [Hoo02], but that view has not gone unchallenged, e.g., [Kin01].

[13] The resemblance persists across multiple accounts of definite descriptions, including Russell's [Rus05], Strawson's [Str71], and Donnellan's [Don66] .

Kotka Radio	All ships in Gulf of Riga, all ships in Gulf of Riga. *This is* Kotka Radio, Kotka Radio. Ice information, area: Gulf of Riga. INFORMATION: The ice type is: winter fast ice, ice change: no change, ice navigation: ice-breaker assistance is necessary

Fig. 2. Distributed definite description example from SeaSpeak

The recommended SeaSpeak message structure is on display in Figure 2. Messages begin with header information and are followed by message elements, each of which begins with one of the message markers described above. Here in Figure 2 we see a header that makes three points:

1. "All ships in Gulf of Riga." This identifies the addressees. The message is being broadcast on an open radio channel.
2. "This is Kotka Radio." This identifies the speaker.
   Note: Identifying the addressee(s) and the speaker has to be done in the header of every message.
3. "Ice information, area: Gulf of Riga." This serves to specify what we call the *context of focus*. The speaker is announcing, declaring, that what follows is
   (a) Information pertaining to ice
   (b) Which ice is in the Gulf of Riga
   Note: Specifying the context of focus occurs in some but not all message headers (and messages).

The INFORMATION then consists of three assertions:

1. *The ice is (of type) winter fast ice.*
2. *The ice has not changed recently.*
3. *Ice-breaker assistance is required for those who wish to navigate the ice.*

In each case, we observe, *the ice* may be interpreted as *the ice in the Gulf of Riga*, producing the following stylistic variants of the three assertions:

1^1. *The ice in the Gulf of Riga is (of type) winter fast ice.*
2^1. *The ice in the Gulf of Riga has not changed recently.*
3^1. *Ice-breaker assistance is required for those who wish to navigate the ice in the Gulf of Riga.*

If *ice* were a singular count term (e.g., *the King of France*, *the man drinking a Martini*), then one's preferred analysis of definite descriptions would be appropriate (see below). In our example, however, *ice* is a mass term. Consequently, we propose the analysis evidenced in the following stylistic variants.

1^2. *There is ice in the Gulf of Riga and it is (of type) winter fast ice.*
2^2. *There is ice in the Gulf of Riga and it has not changed recently.*
3^2. *There is ice in the Gulf of Riga and ice-breaker assistance is required for those who wish to navigate it.*

We note that a stronger interpretation might be preferred:

1^3. *There is ice in the Gulf of Riga and all of it is (of type) winter fast ice.*
2^3. *There is ice in the Gulf of Riga and all of it has not changed recently.*
3^3. *There is ice in the Gulf of Riga and for all of it, if you wish to navigate it, then ice-breaker assistance is required.*

Other variants are possible. In what follows, unless otherwise noted, we employ variant 2.

7 Formalizing Distributed Descriptions into FLBC

Our principal goal in the present section is to combine the example from §5 with the analysis of *the*-expressions from §6. The result will be a structured, generalizable, and formal SeaSpeak message structure, akin to that in Figure 1, but incorporating distributed definite description.

7.1 Definite Descriptions and Nonexistence

Russell's analysis of definite descriptions is an excellent starting place for the method we shall employ here. Briefly, a definite description such as *The King of France is bald* would be analyzed as *There is exactly one thing that is King of France and it is bald.* Formally,

$$\exists x(F(x) \wedge B(x) \wedge \forall y(F(y) \rightarrow x = y)) \tag{1}$$

and similarly for other examples. Russell introduced useful notation for *the x such that* $\phi(x)$, viz., $(\imath x)\phi(x)$. Thus, Expression 1 may be abbreviated as

$$B((\imath x)F(x)) \tag{2}$$

This assumes an axiom schema permitting such formulations:[14]:

$$y = (\imath x)\phi(x) \leftrightarrow \forall x(\phi(x) \rightarrow y = x \wedge \phi(y)) \tag{3}$$

Russell's elegant theory has not gone uncriticized. From Expressions 2 and 3 it follows that $\exists x F(x)$, i.e., that something is King of France. This *might* be appropriate and perhaps in the present example it is. Perhaps, *The King of France is bald* is false just because (or sufficiently because) there is no King of France.

[14] Cf., [Lam91].

There cases in which we wish to speak, and think we can speak truly, about entities that do not or might not exist (in any straightforward way). *The surviving partner in Spade & Archer solved the murder* is—or can be taken as—true, even though we are talking about events in a work of fiction (*The Maltese Falcon* by Dashiel Hammett). If it is not true, then is it false? Such considerations lead naturally to the motivations underlying free logic, the variety of first-order logic in which names are free of existential commitment. Using the standard notation—$E!$—and definition of 'exists'

$$E!(y) \leftrightarrow \exists x(x = y) \tag{4}$$

the axiom schema in Expression 3 may be modified to be conditioned on the existence of the entity in question [Lam91]:

$$E!(y) \rightarrow y = (\imath x)\phi(x) \leftrightarrow \forall x(\phi(x) \rightarrow y = x \wedge \phi(y)) \tag{5}$$

Using expression 5 and given $B((\imath x)F(x))$ it follows that $E!(y) \rightarrow F(y)$, but in free logic, the inference from $F(y)$ to $\exists x F(x)$ is blocked absent $E!(y)$.

We shall steer a course permitting, but not requiring, free logic. We accept Expression 3 as an axiom schema or definition, but we require that in an formula of the form $y = (\imath x)\phi(x)$ that y be instantiated to a name, e.g., a_1. From $a_1 = (\imath x)\phi(x)$ and expression 3 it follows that $\phi(a_1)$ In standard logic this will entail $\exists x\phi(x)$ and in free logic this will entail only $E!(a_1) \rightarrow \exists x\phi(x)$. This affords us the flexibility of either using or discarding free logic as the case mandates.

When would a SeaSpeaker want to employ free logic and names for nonexistent things? It will often be useful to do so. Example: *The next arrival of the Rose Maru is scheduled for tomorrow.* Suppose that the Rose Maru fails to arrive. The sentence might still be true and if not perhaps the speaker is guilty of lying. Free logic will help get the inferences right and is convenient for this purpose.

7.2 Example

Figure 3, page 314, presents an FLBC representation of a SeaSpeak INFORMATION message with distributed definite description used for a *the*-expression. See Figure 2 for the *the*-expression in question. Figure 3 should be compared with Figure 1, page 306 in which a proper name is represented instead of a *the*-expression.

1. Reference-fixing
   s = the speaker
   a = the addressee
2. Presupposition
   (a) a_1 is the Gulf of Riga region:
   $a_1 = (\imath x)\mathit{Location}(x,$ *'Gulf of Riga'*$)$
   (b) a_1 is the focus of context for utterance e_1:
   $\mathit{Focus}(e_1, a_1)$
   (c) There is a state, s_1, of being ice at this time:
   $\mathit{ice}(s_1) \wedge\ \mathit{Hold}(s_1,\ \mathit{now})$
   (d) The state s_1 is located at a_1:
   $\mathit{Location}(s_1, a_1)$
3. Message body
   (a) Propositional content.
   There is ice in the Gulf of Riga and it is of type winter fast ice:
   $\exists x(\mathit{In}(x, s_1) \wedge\ \mathit{Type}(x,$ *winter-fast-ice*$))$
   (b) Illocutionary force.
   Further, the utterance is in the class INFORMATION, which indicates its illocutionary force. SeaSpeak FLBC Sentence 1 represents the INFORMATION force for the utterance to hand, with ϕ serving as a place-holder for the propositional content.
   SeaSpeak FLBC Sentence 4 $\mathit{information}(e_1) \wedge\ \mathit{Speaker}(e_1, s) \wedge\ \mathit{Addressee}(e_1, a) \wedge\ \mathit{Cul}(e_1,\ \mathit{now}) \wedge\ \mathit{Object}(e_1, \ulcorner\phi\urcorner)$

Fig. 3. FLBC representation of a SeaSpeak INFORMATION message with distributed definite description used for a *the*-expression

Points arising:

1. The substantive difference, between the message in Figure 3 (containing a *the*-expression) and the message in Figure 1 (with a proper noun instead of a *the*-expression), lies in section 2, holding the presuppositions.
2. Item (2a) of the message in Figure 3 uses a Russellian definite description to create a name—a_1—for the Gulf of Riga location, to be used as the context for the message.
3. Item (2b) declares that the contextual focus of the e_1 event (keying the *information* illocutionary force verb) is to be a_1, the Gulf of Riga *location* (as distinct from the Gulf of Riga itself).
4. Item (2c) is similar to item (2a) in Figure 1.
5. Item (2d) has no analog in Figure 1. It enforces the declaration of focus in item (2b).
6. With the presuppositions in order, the SeaSpeak FLBC Sentence is quite simple. In fact the illocutionary force components are identical in Figures 1 and 3.

8 Analysis of the Remaining SeaSpeak Message Markers

We have so far given two example analyses for SeaSpeak sentences with the INFORMATION message marker. It remains to discuss the other six message markers.

8.1 SeaSpeak WARNING Message Marker

In general we treat *warning* as an assertion with the presupposition that the speaker thinks there is related danger for the addressee.
Example sentence:

SeaSpeak Sentence 13 *WARNING: The Leading Lights are not lit.*

Stylistic variant:

SeaSpeak Sentence 14 *WARNING: The Leading Lights are unlit.*

Assumption: It is the Leading Lights of the ship Paisano that are not lit.

Figure 4 presents the FLBC message structure (with comments) for our WARNING example.

The points we wish to make about *the*-expressions have now been made. Consequently, for the remaining SeaSpeak message markers we will focus only on the treatment of the markers, eschewing treatment of the full messages, as in Figure 4. This will simplify and, we hope, clarify what follows.

8.2 SeaSpeak INTENTION Message Marker

Example sentence:

SeaSpeak Sentence 15 *INTENTION: I intend to reduce speed, new speed six knots.*

Under the analysis we offer, to say you intend that P amounts to asserting that you will see to it that P. Other analyses are possible and merit investigation, although the one on display is plausible, natural and, we think, quite serviceable. Our analysis, then, suggests the following stylistic variant:

SeaSpeak Sentence 16 *INTENTION: I assert that I will see to it that a speed reduction event occurs, with the new speed being six knots.*

Implicit in the context is that the speaker is referring to the speed of a particular vessel. Let us say it is the Paisano, a_1, as in Figure 4. Here, e_3 will be a *speed-reduction* event, whose *Theme* and *Focus* is a_1, the Paisano. What it is that is supposed to be seen to, then, is

1. Reference-fixing
   s = the speaker
   a = the addressee
   a_1 = *Paisano* (the registered ship of that name)
   LeadingLights(a_2) (a_2 has the property of being Leading Lights.)
2. Presupposition
   (a) The theme or topic of this message is a_2:
   Theme(s_1, a_2)
   (b) The focus of context for utterance s_1 is a_1:
   Focus(s_1, a_1)
   (c) The theme a_2 is located at a_1:
   Location(a_2, a_1)
   (d) The s_1 state is dangerous for a, the addressee:
   dangerous$(s_1) \wedge$ *Benefactive*(s_1, a)
3. Message body
   (a) Propositional content.
   a_2 (having the property of being Leading Lights located on the Paisano) is unlit:
   $(\mathit{unlit}(s_1) \wedge \mathit{In}(a_2, s_1))$
   (b) Illocutionary force.
   Further, the utterance is in the class WARNING, which indicates its illocutionary force is *assert*.
   SeaSpeak FLBC Sentence 5 *assert*$(e_1) \wedge$ *Speaker*$(e_1, s) \wedge$ *Addressee*$(e_1, a) \wedge$ *Cul*$(e_1,$ *now*$) \wedge$ *Object*$(e_1, \lceil$ $(\mathit{unlit}(s_1) \wedge \mathit{In}(a_2, s_1))$ $\rceil)$

Fig. 4. FLBC representation of a SeaSpeak WARNING message with a *the*-expression

SeaSpeak FLBC Sentence 6 *reduce-speed*$(e_3) \wedge$ *Theme*$(e_3, a_1) \wedge$ *Cul*$(e_3,$ *now*$)$

Seeing to it that is represented by the *stit* predicate, the FLBC analog of the *stit* operator in action logic.[15] The FLBC Stit Schema—

FLBC Schema 1 (FLBC Stit Schema) $\mathit{stit}(e_1) \wedge \mathit{Agent}(e_1, j) \wedge \mathit{Cul}(e_1, t_1) \wedge \mathit{Content}(e_1, \lceil \phi \rceil)$

—is rendered into English as *Agent j sees to it at time t_1 that ϕ.* Thus, adding the *stit* aspect to the speed reduction we get:

SeaSpeak FLBC Sentence 7 $\mathit{stit}(e_2) \wedge \mathit{Agent}(e_2, s) \wedge \mathit{Cul}(e_2, \mathit{now}) \wedge \mathit{Content}(e_2, \lceil \mathit{reduce\text{-}speed}(e_3) \wedge \mathit{Theme}(e_3, a_1) \wedge \mathit{Cul}(e_3,$ *now*$) \rceil)$

on the assumption that the speaker is s. Finally, we wrap all of this in an assertion to get:

[15] See discussion in [JK04], this volume, also [KY04].

SeaSpeak FLBC Sentence 8 $assert(e_1) \wedge\ Speaker(e_1, s) \wedge\ Addressee(e_1, a) \wedge\ Cul(e_1,\ now) \wedge\ Object(e_1, \lceil\ stit(e_2)\ \wedge\ Agent(e_2, s)\ \wedge\ Cul(e_2, now)\ \wedge\ Content(e_2, \lceil reduce\text{-}speed(e_3) \wedge\ Theme(e_3, a_1) \wedge\ Cul(e_3,\ now)\ \rceil)\ \rceil)$

8.3 SeaSpeak REQUEST Message Marker

The SeaSpeak manual [WGJS88, page 97] has this to say about requests:

> The word REQUEST will be used to signal messages which mean 'I want something to be arranged or provided' as in ships' stores requirements, bunkering, permission, ... It is commonly accompanied by the word **Please**, e.g.
>
> REQUEST: Please send, quantity: five acetylene cylinders.

SeaSpeak REQUESTs work, we think, more or less as ordinary requests as treated in the speech act literature. More specifically, a request differs from a command in that the request does not invoke any extraordinary authority. Recall the dialog in the movie *Casablanca*.

```
                LASZLO
Captain Renault, I am under your
authority.  Is it your order that we
come to your office?

                RENAULT
     (amiably)
Let us say that it is my request.
That is a much more pleasant word.

                LASZLO
Very well.
```

(See http://www.geocities.com/classicmoviescripts/ for a link to the script.)

Letting ϕ represent the propositional content of a REQUEST, we model the illocutionary force as follows:

SeaSpeak FLBC Sentence 9 $request(e_1) \wedge\ Speaker(e_1, s) \wedge\ Addressee(e_1, a) \wedge\ Cul(e_1,\ now) \wedge\ Object(e_1, \lceil\ \phi\ \rceil)$

We note that in the above example—*Please send, quantity: five acetylene cylinders*—it is not specified to whom the cylinders should be sent. Under the FLBC representation this would be indicated (or not) in the propositional content, ϕ, or in the presupposition. If, for example, the addressee is to send the material to the speaker, the content would include this fragment: $send(e_1) \wedge\ Agent(e_1, a)$ and the presupposition could include $Benefactive(e_1, s)$, perhaps by default.

8.4 SeaSpeak ADVICE Message Marker

The SeaSpeak manual [WGJS88, page 97] has this to say about the ADVICE marker:

> The word ADVICE will be used to signal suggestions, e.g.
>
> ADVICE: Anchor, position: bearing: one-nine-four degrees true, from Keel Point distance: one mile.

An advice, then, is a suggestion. We broadly agree with Searle and Vanderveken [SV85, page 202] that a suggestion in this sense is a kind of request (their category is *directive*). As we model them, suggestions (and SeaSpeak ADVICEs) are requests with the presumption that the beneficiary is the addressee. Thus, ADVICE comes with the presupposition that the event keying the propositional content, say e_2, benefits the addressee, a: $\mathit{benefits}(e_2, a)$. Note that *Benefactive* is a thematic role and does not imply anything good (or bad) for anyone. Even in ordinary English we can say "Brenda was the beneficiary of his tongue lashing and assorted insults" without suggesting there was anything good for Brenda in this. The special predicate $\mathit{benefits}(x, y)$ is to be interpreted as indicating a good of some kind, that y benefits (would benefit) from x occurring. Further, this is a general presupposition, unlike the referential examples above, in that it is not specific to the message to hand. Thus, it may be given as a general rule or axiom schema as part of the interchange agreement governing the communications. For development of this idea see [JK04] in this volume.

8.5 SeaSpeak INSTRUCTION Message Marker

The SeaSpeak manual [WGJS88, page 97] has this to say about the INSTRUCTION marker:

> The word INSTRUCTION will be used to signal commands, e.g.
>
> INSTRUCTION: Go to berth number: two-five.

We analyze commands as requests (or directives) in which the speaker invokes an authority and, if all goes well, puts the addressee under an obligation to honor the request. (See [JK04] in this volume for elaboration.) However, that there is an authority invoked and that the addressee is under an obligation need not be explicitly stated in the message. These are general conditions and they should be inferred from general rules governing the conversation. Consequently, our representation task here (and the burden on the communicants) is much reduced.

Even so, it will helpful for the speaker to declare the authority he is claiming. A simple and workable way to do this is to employ a special predicate,

AuthorityTitle(x, y, z),
with interpretation *x has and invokes the authority of y for the sake of z.* Thus, for example, *AuthorityTitle*(*s, 'Lighthouse Master',* e_1) added to the presupposition section would declare that speaker s is invoking the authority of a Lighthouse Master in uttering the messaged keyed by e_1. The illocutionary force of that message would have the form:

SeaSpeak FLBC Sentence 10 *command*$(e_1) \wedge$ *Speaker*$(e_1, s) \wedge$ *Addressee*$(e_1, a) \wedge$ *Cul*$(e_1,$ *now*$) \wedge$ *Object*$(e_1, \ulcorner \phi \urcorner)$

8.6 SeaSpeak QUESTION Message Marker

The SeaSpeak manual [WGJS88, page 97] has only this to say about the QUESTION marker:

> The word QUESTION will be used to signal all questions, e.g.
>
> | QUESTION: What is your ETA at the dock entrance? |
> |---|

Thus, the SeaSpeak QUESTION message marker is for the most part unproblematically interpreted as indicating a question in the usual, ordinary language sense.

We model questions as reference-fixing assertions recognized as presuppositions, plus requests to describe the referents. For example, a stylistic variant of "What time is it?" would be "There is some time that it is. [assertion presupposed] Please describe that time to me. [a request]" Similarly, "Senator, when did you stop beating your wife?" is a stylistic variant of "You used to beat your wife. Please describe when it was the beating stopped." Thus, a question (typically) makes a presupposition and goes bad if the presupposition is false or otherwise problematic. This is true of both w-questions (involving *who, what, where, when,* and *why*) and yes-no questions (answered with a 'yes' or a 'no'). We will examine one example of each kind.

W-Questions. Consider the what-question:

SeaSpeak Sentence 17 *QUESTION: What is your position?*

As usual, let s be the speaker and a the addressee. On our analysis, the speaker presumes that there is exactly one position that the addressee has, then requests that the addressee describe it. Thus, in the presupposition section of the message we would find

SeaSpeak FLBC Sentence 11 $x_1 = (\imath x)($*location*$(x) \wedge$ *At*$(a, x) \wedge$ *Hold*$(x,$ *now*$))$

which posits x_1 as a's one and only location. Additionally, s would state the presumption that x_1 exists: $E!(x_1)$.

The question is completed by a SeaSpeak (FLBC) sentence in which s requests that a describe x_1, a's one and only location.

SeaSpeak FLBC Sentence 12 $\mathit{request}(e_1) \wedge\ \mathit{Speaker}(e_1, s) \wedge\ \mathit{Addressee}(e_1, a) \wedge\ \mathit{Cul}(e_1,\ \mathit{now}) \wedge\ \mathit{Object}(e_1, \lceil\ \mathit{describe}(e_2) \wedge\ \mathit{Agent}(e_2, a) \wedge\ \mathit{Theme}(e_2, x_1) \wedge\ \mathit{Benefactive}(e_2, s) \wedge\ \mathit{Cul}(e_2,\ \mathit{now})\ \rceil)$

Yes-No Questions. Consider the yes-no question:

SeaSpeak Sentence 18 *QUESTION: Are you going to pass me soon?*

As a presupposition, the speaker expresses the event being asked about—

SeaSpeak FLBC Sentence 13 $\mathit{passing}(e_3) \wedge\ \mathit{Agent}(e_3, a) \wedge\ \mathit{Theme}(e_3, s) \wedge\ \mathit{Hold}(e_3,\ \mathit{soon})$

—a passing of s by a soon. The speaker does *not* presume that the event e_2 exists. That is the information requested in the SeaSpeak sentence:

SeaSpeak FLBC Sentence 14 $\mathit{request}(e_1) \wedge\ \mathit{Speaker}(e_1, s) \wedge\ \mathit{Addressee}(e_1, a) \wedge\ \mathit{Cul}(e_1,\ \mathit{now}) \wedge\ \mathit{Object}(e_1, \lceil\ \mathit{describe}(e_2) \wedge\ \mathit{Agent}(e_2, a) \wedge\ \mathit{Theme}(e_2, e_3) \wedge\ \mathit{Benefactive}(e_2, s) \wedge\ \mathit{Cul}(e_2,\mathit{now})\ \rceil)$

Because the speaker has not presumed $E!(e_3)$, i.e., that e_3 actually exists, the addressee is to infer by the conventions governing the conversation that a proper answer consists of asserting either that $E!(e_3)$ (the addressee will pass soon) or $\neg E!(e_3)$. Questions, we believe, inherently involve presuppositions, and when we presuppose about events and things that might not exist, free logic is a most handy tool.

9 Discussion and Conclusion

SeaSpeak, we noted at the outset, affords a natural experiment of sorts. It was conceived and developed without any apparent regard for doctrines of speech act theory or for suitability to formalization by an ACL (agent communication language). This and the fact that SeaSpeak is successful—it is used and entrenched in maritime communications—presents an apt challenge for particular ACLs and, more generally, the programme of formalization of business communication.

At least in a preliminary and theoretical way this challenge has been met by FLBC. First, we note that the message marker structure in SeaSpeak messages can only be seen as a heartening confirmation of a basic tenet of speech act theory, the $F(P)$ framework. Second, in two previous papers [KLPY03,KY04] we have explored with some success the translation

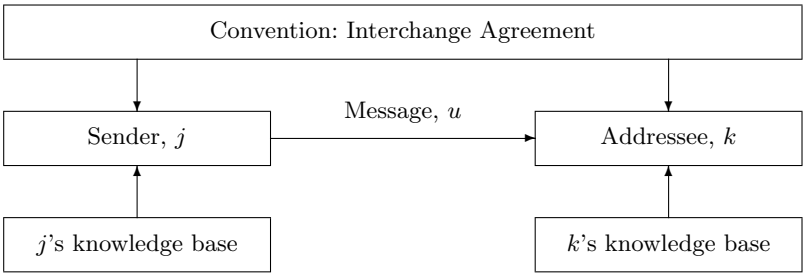


Fig. 5. Basic Messaging Framework: Message u from speaker j to addressee k (after [KT00])

of SeaSpeak sentences into FLBC, without at the same time addressing the problem of dynamic reference fixing. In the present paper, we have addressed this problem (at least in principle) and we have extended the analysis of SeaSpeak sentences to entire SeaSpeak messages. Significantly, these two issues are tied together by a common use of presupposition. In the case of *the*-expressions, we find a presupposition of a context of focus. In the case of questions, we find them to be inseparable from their presuppositions.

What about practical considerations? Is there any prospect in all of this for deployed applications? We think there is, but absent a working demonstration we opt not to stretch the reader's credulity (or our credibility). Nevertheless, we close with two observations. First, automated messaging occurs within an institutional context. See Figure 5 for a framework. That context includes conventions governing the exchange and interpretation of messages. Businesses using EDI even have a name for (a part of) this convention: the interchange agreement. It is the interchange agreement (or its equivalent) that determines, for example, how many message markers there are and what they mean. Our analysis shows that in SeaSpeak significant parts of a message belong to the reference-fixing or presupposition sections. These elements may often, then, be supplied by default, as we have remarked, because they are specified by a governing convention. In principle, what this means for practice is that with careful formalization *and* well-crafted conventions the information burden on the message composer can be greatly reduced. Messages may be 'lean' [KT00] and still convey all the essential information. Whether one is designing a user interface for humans or a messaging-generating program, this is welcome news.

Our second observation has to do with reference fixing. Many, if not most, of the problematic uses of *the*-expressions and related indexical reference-fixing devices can be handled as we have shown by fixing a context of focus and then relying on a (distributed) description. In ordinary language conversation this fixing of context is most often done by pointing in some way or by using a demonstrative expression (involving *this, that, these,* and *those*).

Is there a way for a human to instruct a machine in this way? How might a machine discern the object of an indexical by a human? In the simple case in which the context of focus has a proper name, e.g., the ship Paisano, there is little difficulty. The human interface is designed to capture the intended focus (say by picking from a list of names) and the computer program is given an established list of names and information about their objects. This simple case, we observe, may be generalized. Instead of mutually working with a common list of names, the communicants might be given maps or diagrams. A speaker might point (in any of various ways) to a region of a map and a supportive procedure could (for a suitably represented map) automatically extract the pointed-to context of focus. This might be done if the 'speaker' is a computer process and the addressee is a human, or vice versa.

These observations are, we think, straightforward enough. The consequences for applications can only be speculative at this point. We hope others will join us in exploring these possibilities.

References

[AEB04] Alan S. Abrahams, David M. Eyers, and Jean M. Bacon, *Practical contract storage, checking, and enforcement for business process automation*, Formal Modelling for Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, Berlin, Germany, 2004.

[Aus62] John L. Austin, *How to do things with words*, Oxford at the Clarendon Press, Oxford, England, 1962.

[Ben03] John Benyon, *Building police co-operation: The European construction site around the third pillar*, World Wide Web file, Accessed 28 February 2003, Dated 1996. `http://www.psa.ac.uk/cps/1996/beny.pdf`.

[BH79] Kent Bach and Robert M. Harnish, *Linguistic communication and speech acts*, The MIT Press, Cambridge, Massachusetts, 1979.

[BM85] David C. Blair and M. E. Maron, *An evaluation of retrieval effectiveness for a full-text document-retrieval system*, Communications of the ACM **28** (1985), no. 3, 289–299.

[Don66] Keith S. Donnellan, *Reference and definite descriptions*, The Philosophical Review **75** (1966), no. 3, 281–304.

[FBH86] Eileen Fitzpatrick, Joan Bachenko, and Don Hindle, *The status of telegraphic sublanguages*, Analyzing Language in Restricted Domains: Sublanguage Description and Processing (Ralph Grishman and Richard Kittredge, eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1986, pp. 39–51.

[Fos82] Antony Charles Foskett, *The subject approach to information*, 4th ed., Linnet Books, Hamden, CT, 1982.

[Har68] Zellig Harris, *Mathematical structures of language*, John Wiley & Sons, New York, NY, 1968.

[Har02] Richard K. Harrison, *Bibliography of planned languages (excluding Esperanto)*, World Wide Web, 1992-2002, accessed December 2002, `http://www.invisiblelighthouse.com/ langlab/bibliography.html`.

[Hoo02] Christopher Hookway, *Truth, rationality, and pragmatism: Themes from peirce*, Clarendon Press, Oxford, UK, 2002.

[J+93] Edward Johnson et al., *Policespeak: Police communications and language, English-French lexicon*, PoliceSpeak Publications, Cambridge Research Laboratories, 181a Huntingdon Road, Cambridge, CB3 ODJ, 1993, ISBN: 1 898211 01 9.

[Jac02] Ray Jackendoff, *Foundations of language: Brain, meaning, grammar, evolution*, Oxford University Press, Oxford, UK, 2002.

[JK04] Andrew J.I. Jones and Steven O. Kimbrough, *A note on modeling speech acts as signalling conventions*, Formal Modelling for Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, Berlin, Germany, 2004.

[Joh98] Edward Johnson, *LinguaNet final report*, World Wide Web file, 1998, http://www.hltcentral.org/usr_docs/project-source/linguanet/Final-Report/Final-Report.html, accessed 28 February 2003.

[Joh02] ———, *Talking across frontiers*, Proceedings of the International Conference on European Cross-Border Co-operation, 2002, Available at: http://www.prolingua.co.uk/talking.pdf. Accessed December 2002.

[Kim99] Steven O. Kimbrough, *Formal language for business communication: Sketch of a basic theory*, International Journal of Electronic Commerce **3** (Winter 1998–99), no. 2, 23–44.

[Kim90] ———, *On representation schemes for promising electronically*, Decision Support Systems **6** (1990), no. 2, 99–121.

[Kim01] ———, *Reasoning about the objects of attitudes and operators: Towards a disquotation theory for representation of propositional content*, Proceedings of ICAIL '01, International Conference on Artificial Intelligence and Law, 2001.

[Kim02] ———, *A note on the Good Samaritan paradox and the disquotation theory of propositional content*, Proceedings of ΔEON'02, Sixth International Workshop on Deontic Logic in Computer Science (John Horty and Andrew J.I. Jones, eds.), May 2002, pp. 139–148.

[Kin01] Jeffrey C. King, *Complex demonstratives: A quantificational account*, MIT Press, Cambridge, MA, 2001.

[KK01] Kyra Karmiloff and Annette Karmiloff-Smith, *Pathways to language: From fetus to adolescent*, Harvard University Press, Cambridge, MA, 2001.

[KLPY03] Steven O. Kimbrough, Thomas Y. Lee, Balaji Padmanabhan, and Yinghui Yang, *On original generation of structure in legal documents*, Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL), 2003.

[KM97] Steven O. Kimbrough and Scott A. Moore, *On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity*, ACM Transactions on Information Systems **15** (October 1997), no. 4, 321–367.

[KT00] Steven O. Kimbrough and Yao-Hua Tan, *On lean messaging with unfolding and unwrapping for electronic commerce*, International Journal of Electronic Commerce **5** (2000), no. 1, 83–108.

[KY04] Steven O. Kimbrough and Yinghui Yang, *Action at the tables: Sketching a tabular representation for utterances under the language–action perspective*, Proceedings of the 9th International Working Conference on the

Language Action Perspective on Communication Modelling (Rutgers, The State University of New Jersey, New Brunswick, NJ) (Mark Aakhus and Mikael Lind, eds.), School of Communication, Information, and Library Studies, 2–3 June 2004, http://www.scils.rutgers.edu/˜aakhus/lap/lap04.htm, pp. 103–120.

[Lam91] Karel Lambert (ed.), *Philosophical applications of free logic*, Oxford University Press, Oxford, U.K., 1991.

[LAP04] LAP, *Home LAP '04*, World Wide Web page, September 2004, http://www.scils.rutgers.edu/∼aakhus/lap/lap04.htm.

[Lar85] Andrew Large, *The artificial language movement*, Basil Blackwell, New York, NY, 1985, ISBN 0-631-14497-8.

[Lev83] Stephen C. Levinson, *Pragmatics*, Cambridge University Press, Cambridge, England, 1983.

[Lin03] LinguaNet, *The linguanet project*, 2003, www.cbs.dk/departments/fir/-linguanet/, accessed 2003-02-28.

[Mac30] T. C. Macaulay, *Interlanguage*, The Clarendon Press, Oxford, UK, 1930.

[Ogd38] C. K. Ogden, *Basic English : A general introduction with rules and grammar*, K. Paul, Trench, Trubner, London, UK, 1938.

[Pro03] Prolingua, *Operational communications, controlled languages, computing*, World Wide Web page, Accessed 28 February 2003, www.prolingua.co.uk/.

[Ric43] I. A. Richards, *Basic English and its uses*, W. W. Norton & Company, Inc., New York, NY, 1943.

[Rus05] Bertrand Russell, *On denoting*, Mind (1905), 479–93, Available at: http://www.mnstate.edu/gracyk/courses/web%20publishing/russell_on_-denoting.htm (accessed 30 July 2004).

[Sag75] Naomi Sager, *Sublanguage grammers [sic] in information processing*, Journal of the American Society for Information Science **26** (1975), no. 1, 10–16.

[Sag86] ———, *Sublanguage: Linguistic phenomenon, computational tool*, Analyzing Language in Restricted Domains: Sublanguage Description and Processing (Ralph Grishman and Richard Kittredge, eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1986, pp. 1–17.

[Sea69] John R. Searle, *Speech acts*, Cambridge University Press, Cambridge, England, 1969.

[Str71] P. F. Strawson, *Logico-linguistic papers*, ch. On Referring, Methuen, 1971, Originally published 1957.

[SV85] John R. Searle and Daniel Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, Cambridge, England, 1985.

[Swa80] Charles Swanland, *Basic English for science & technology*, Intercultural Press, Chicago, IL, 1980.

[WGJS88] Fred Weeks, Alan Glover, Edward Johnson, and Peter Strevens, *Seaspeak training manual: Essential English for international maritime use*, Pergammon Press, Oxford, UK, 1988, Available via www.maritimeusa-.com.

A Note on Modelling Speech Acts as Signalling Conventions

Andrew J.I. Jones[1] and Steven Orla Kimbrough[2]

[1] King's College, London, UK,
ajijones@dcs.kcl.ac.uk
[2] University of Pennsylvania, Philadelphia, PA, USA,
kimbrough@wharton.upenn.edu

Abstract. This paper presents a fully formal integration of Jones's logical theory of speech acts as signalling conventions with Kimbrough's Formal Language for Business Communication (FLBC). The work is part of a larger programme of logicism in the context of electronic commerce. Speech acts are an especially apt subject for this programme because of their pervasiveness and importance in communication for all commerce, electronic or not. The paper demonstrates that the conventionist view of speech acts, embodied in Jones's logical theory, fits naturally with Kimbrough's FLBC and with the Basic Messaging Framework for business communications. Further, the paper provides an illustration of how the resulting integrated theory might be implemented in practice through logic programming.

1 Introduction

A logicist may hold any of several views on the role and value of formal logic in electronic commerce. Prominent among these views are:

- Formal logic is a useful, perhaps even preferred, tool for analyzing and clarifying concepts of import in electronic commerce.
- Formal logic is a useful, perhaps even preferred, tool for articulating important kinds of specifications pertaining to electronic commerce. Among these kinds are specifications for designing ma-chine-to-machine messaging systems.
- Logic in the applied form of logic programming is potentially a valuable, perhaps even preferred, vehicle for implementation of machine-to-machine messaging systems.

We are logicists, at least in the context of electronic commerce, and we believe that there is much to be said in favor of each of these views. Too much in fact to fit into a short paper. Our present ambitions are more limited. We aim to sketch a formal and logical theory of speech acts as conventional signalling acts. In virtue of being formal the theory affords rigorous, machine-readable representation. In virtue of being logical a well-defined and justified formal inference apparatus is part of the theory. Our strategy for constructing this

theory is to combine Kimbrough's FLBC theory[1] with Jones's theory of conventional signalling acts and its attendant logical framework.[2] Kimbrough's FLBC theory is a representational theory, in first-order logic, that is apt for expressing signalling acts (among other things). It is not, however, an account of what signalling acts are, of what constitutes them. Jones's theory of conventional signalling acts is such an account. With proper attention to details, the two theories fit together hand in glove, as we shall explain.

Speech acts are of fundamental import and enduring interest for electronic commerce, and generally for understanding language and communication. The concept of the speech act is well entrenched in a number of disciplines, including linguistics, philosophy, the computational sciences generally, and particularly in the thinking of researchers in electronic commerce. The underlying notion—originating with Austin [Aus62] and further developed by Searle [Sea69] and others—is that speaking is a kind of doing or acting, and that we should consider the broad range of kinds of things that agents can do with words, rather than one-sidedly focusing on acts of stating that such-and-such is the case.

Speech acts are interesting theoretically because they seem to be so pervasive in language, and because of the logical and conceptual challenges in developing a workable formal theory of them. These two factors also motivate the practical interest in speech acts, evidenced by researchers in electronic commerce. Pervasive in commercial transactions are mundane communications—purchase orders, invoices, receiving reports, etc.—that are required in great volume and that should be, all agree, very profitable targets for formalization and automation. These communications, it is broadly agreed, are properly viewed as cases of speech acts. To issue an invoice is (roughly) to request payment for goods received. To issue a receiving report is (roughly) to assert that such-and-such goods have arrived in good condition. To issue a purchase order is (roughly) to request that ownership of certain goods be transferred to the speaker, in consideration of which the speaker promises to pay the current owner a certain amount of money.

It is a handicap to electronic commerce not to have an adequate approach to formalization of speech acts. Our longer-term goal is to replace that deficiency with a productive, well-founded, formal and implementable theory. This note is meant as a step in that direction.

2 Asserting: Two Prototypes

Our purpose in this paper is to demonstrate the coherence, indeed the felicity, of combining Kimbrough's FLBC with Jones's theory of conventional signalling acts. In the interests of ease and clarity of exposition, we will proceed incrementally, and we begin with a discussion of asserting.

[1] E.g., [Kim99], [KM97], [KT00], [Kim01], and [Kim02].
[2] E.g., [Jon02,Jon04,JP04].

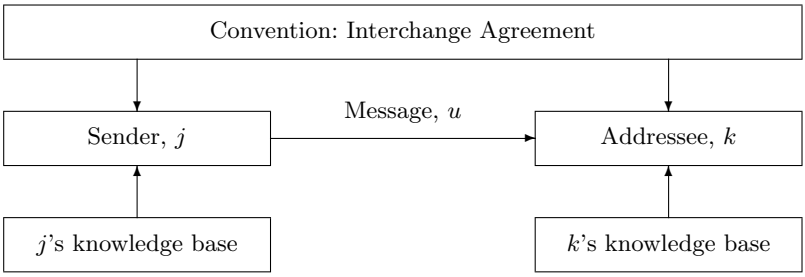


Fig. 1. Basic messaging framework: message u from speaker j to addressee k (after [KT00])

FLBC is a formalism for representing speech act messages. It succeeds to the extent that it affords articulation of speech acts and their contents, and as well as proper inferencing upon them. FLBC is not, and is not intended to be, a formal theory of the logic of speech acts. Instead, it is intended to fit with, to be workable with, a logical theory of speech acts. Jones's account of conventional signalling systems is just such a theory. Put in terms of the Basic Messaging Framework, Figure 1, FLBC is about structuring the messages, u, and Jones's theory of conventional signalling systems belongs to the Interchange Agreement. FLBC is about how we say what we want to say; Jones's theory is about how we may make inferences on what is said.

To an approximation adequate for present purposes, any speech act, and in particular any message u, may be decomposed into an illocutionary force, F, and its propositional content, C. We express them jointly as $F(C)$. The notation is for convenience of exposition; it is a framework and is not part of a logic. That will come shortly.

Jones's core formula in the case of assertion is

Expression 1 (Jones's Assert Schema) $E_j U \Rightarrow_s I_s^* C$

(See [Jon04,JP04].) We have made a change of variables to suit present purposes.) Rendered into English, Expression 1 says that j's seeing to it (E_j) that U counts as making it the case in conventional signalling system s ($\Rightarrow_s$) that were s in an optimal state with respect to its function of facilitating the reliable transmission of reliable information, C would be true. Seeing to it that (E_j), counts as ($\Rightarrow_s$), and ideal functioning (I_s^*) each have their own logics, which we will not discuss in any detail, since they are treated elsewhere.[3] Points arising now:

1. U in Jones's theory may be any (description of a) state of affairs brought about by agent j and falling under the system of conventional signalling, s. U may be the waving of a flag, the shrugging of a shoulder, or whatever

[3] See [Jon04,JP04,JS96].

is stipulated by the convention in force, so long as it is described by the proposition U.
2. The class of signalling conventions in which we are interested here are those that use FLBC expressions to perform U's role.
3. A philosophical point, to which we shall return in comment 4c on Axiom Schema 2, below page 329, is that there is in Expression 1 no requirement of any intention on the part of j, or anyone else. Expression 1 says that—by convention in signalling system s—j's bringing it about that U is a means of stating that C.

FLBC should be seen as (belonging to) a particular convention for undertaking speech acts by machine. According to this convention—let us call it f—one, j, asserts that C to k by sending a message to k of the form:

Expression 2 (FLBC Assert Schema) $\mathit{assert}(e_1) \wedge \mathit{Speaker}(e_1, j) \wedge \mathit{Addressee}(e_1, k) \wedge \mathit{Cul}(e_1, t_1) \wedge \mathit{Content}(e_1, \ulcorner C \urcorner)$

Rendered into English, Expression 2 says that there is an asserting event, e_1, whose speaker is j, and whose propositional content is C. Further, k is to whom the assertion is directed and the assertion event happened (culminated) at time t_1. Note that e_1, j, k, and t_1 are place holders for particular names, i.e., for particular logical constants which are supplied in any given instantiation. Similarly, C is a place holder for a particular formula, to be supplied in any given instantiation. Under f, the *FLBC Speech Act Convention* for communication in a Basic Messaging Framework, j's sending a message, u (see Figure 1), having the form of Expression 2 counts as j's asserting to k that C.

Accompanying Expression 2 in FLBC is an axiom schema formalizing veridicality of an assertion.

Axiom Schema 1 (FLBC Assert Axiom Schema)
$(\mathit{assert}(e) \wedge \mathit{Content}(e, \ulcorner C \urcorner)) \rightarrow (\mathit{Veridical}(e) \leftrightarrow C)$

This belongs to f and is part of the Interchange Agreement.

FLBC and Jones's theory of speech act signalling conventions are now easily combined by adding the following expression to the Interchange Agreement between the communicating parties.

Axiom Schema 2 (Governing Assert Speech Act Axiom Schema)
$E_j(\mathit{assert}(e) \wedge \mathit{Speaker}(e, j) \wedge \mathit{Addressee}(e, k) \wedge \mathit{Cul}(e, t) \wedge \mathit{Content}(e, \ulcorner C \urcorner)) \Rightarrow_f I_f^* C$

Points arising:

1. Axiom Schema 2 is a special case of Expression 1.
2. Axiom Schema 2 should be seen as belonging to the Interchange Agreement. Again, e_1, j, k, and t_1 are place holders for particular names, i.e., for particular logical constants which are supplied in any given instantiation.
3. Successful messaging—utterance of an assert speech act to the effect that C—works as follows under f.
   (a) Parties j and k agree to convention f.
   (b) j sends a message (u in Figure 1) to k, having the form of Expression 2 and using a method of authentication agreed to in f.
   (c) k receives the message, u, validates its authenticity (i.e., that it is indeed from j), and (under an obligation stipulated by f) concludes that $E_j u$. Further, by the logic of seeing to it that (of which more below), k also concludes that u.
   (d) From $E_j u$ and Axiom Schema 2, k is also licensed to conclude that $I_f^* C$.
4. There are a number of ways in which messaging in this context may be unsuccessful or infelicitous in some way. These include the following:
   (a) The message u could be ill-formed with regard to f.
   Messages in FLBC are fully formal and can be exactly specified. These specifications should belong to f, as well as rules for handling violations. An advantage of logicism and full formalization here is that acceptance or rejection of messages can be specified rigorously and automated.
   (b) The content asserted by the speaker, C, might not be believed by the speaker, who may know that its denial is true.
   People lie and no logical system will, or should, prevent that. If $\neg C$, then it will follow from Axiom Schema 1 and the truth of its antecedent that $\neg V(e_1)$, where e_1 is the message ID of j's original utterance.
   (c) The speaker, j, may disavow the assertion.
   The authentication system in place, which must be part of the governing convention, is crucial. If it is easily defeated, the signalling system is unlikely to be successful for very long. If the authentication system is reliable, there remains the problem of distinguishing accidental utterances from fraudulent attempts to renege. The default assumption is that if an f-message is transmitted, the act of transmission counts as an instance of implementation of the governing signalling convention, and so *means what*—according to that convention—*it says.* If the sender (or its owner) wishes to maintain that the transmission was made in error, then the onus of proof will be on the sender or owner to show just that. (Perhaps the governing legal system, or some other relevant authority, will be the adjudicator of last resort on the

issue of distinguishing between errors and genuine signalling acts.) In the case of human signallers, some speech-act theorists would appeal, at this point, to *intention,* to sort the act-tokens that count as literal implementation of a governing signalling convention from those that do not so count. But the view taken in this paper is that there is here no *need* to resort to intention. A good thing this, in the context of e-commerce, since it is very unclear what ascription of intentions to electronic agents amounts to; and equally obscure, therefore, is the issue of how to specify the empirical conditions that would have to be satisfied before an electronic agent could be said to *have* a particular intention. But communicate they can, these electronic agents, for they can perform acts which—according to the governing interchange agreement—have a meaning.

(d) Disputes arise because messages are ambiguous.
This is always a possibility, as is resolution by adjudication. That said, it must also be admitted that it would be difficult to find a more transparent and clear form of formalization than logic as on display here. The *prima facie* case for clarity is strong.

Following a valid assert message (whether felicitous or not), the recipient, k, is in a good position to make automated inferences of import, depending of course on the actual course of events. As noted by Jones for example,[4] our present expressions may be combined with a belief operator B_j (j believes that) to articulate various *positions* of belief and trust. Again, in the interests of brevity, we refer the reader to the original discussions [Jon02,Jon04,JP04].

3 Other Speech Acts

While there are an indefinitely large number of distinct speech act types, most authors recognize a core group that includes assertions, commissives, requests, and declaratives. We will limit our discussion largely to these. The pattern we saw in the case of asserting persists, and will (we believe) persist for other types of speech acts.

3.1 Commands & Requests

Expression 3 (Jones's Command Schema) $E_jU \Rightarrow_s I_s^* O E_k C$

In stylized English, if U counts as a command (by j to k), then in all circumstances in which the conventional signalling system, s, is working ideally (I_s^*), it is obligatory (O) that k sees to it (E_k) that C. The thought is that it is constitutive of the concept of a command that, if given felicitously, there is an obligation created for the one commanded to do what was commanded.

[4] [Jon02,Jon04,JP04]

Jones's theory views requests as weaker than, or at least as somewhat different from, commands.

Expression 4 (Jones's Request Schema) $E_jU \Rightarrow_s I_s^* H_j E_k C$

In English, the upshot of a felicitous request is that the requester, j, attempts to see to it (H_j) that k sees to it that C.

FLBC has historically taken the view that both commands and requests (in Jones's sense) are varieties of the covering speech act, request, and are characterized by being honored or not, just as assertions are either veridical or not. In FLBC, then, we have:

Expression 5 (FLBC Command Schema) $command(e_1) \wedge Speaker(e_1, j) \wedge Addressee(e_1, k) \wedge Cul(e_1, t_1) \wedge Content(e_1, \lceil C \rceil)$

and

Expression 6 (FLBC Request Schema) $request(e_1) \wedge Speaker(e_1, j) \wedge Addressee(e_1, k) \wedge Cul(e_1, t_1) \wedge Content(e_1, \lceil C \rceil)$

along with

Axiom Schema 3 (FLBC Request Axiom Schema) $(request(e) \wedge Content(e, \lceil C \rceil)) \rightarrow (Honored(e) \leftrightarrow C)$

and

Axiom Schema 4 (FLBC Command Axiom Schema) $(command(e) \wedge Content(e, \lceil C \rceil)) \rightarrow (Honored(e) \leftrightarrow C)$

Following the pattern in the case of asserting, this leads directly to:

Axiom Schema 5 (Governing Command Axiom Schema) $E_j(command(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Cul(e, t) \wedge Content(e, \lceil C \rceil)) \Rightarrow_f I_f^* O E_k C$

and

Axiom Schema 6 (Governing Request Axiom Schema) $E_j(request(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Cul(e, t) \wedge Content(e, \lceil C \rceil)) \Rightarrow_f I_f^* H_j E_k C$

Points arising:

1. If, as in Jones's view, commands are rather different from requests, grouping them together as acts that may or may not be honored, as in the FLBC schemata above, may not be appropriate; finer distinctions may be called for.

2. Obligations may themselves be relativized to institutions. Contrast, for example, a command from a policeman and a command from an arbiter of social etiquette. Thus, finer distinctions may be appropriate for the analysis of commands. We note that the counts as operator ($\Rightarrow_s$) is indexed by institution as is the ideality operator (I^*_s). These indices might also be used for relativizing obligations to institutions.
3. As for requests, one way of attempting to see to it that someone does something is to place that person under an obligation to do so.
4. The upshot here is that while different analyses are certainly possible the associated logical apparatus is quite flexible and will support a broad range of views.

3.2 Commissives

Promises and other kinds of commissives have the function, in Jones's analysis, of placing the speaker under an obligation to see to it that what was promised (or committed to) comes about.

Expression 7 (Jones's Commissive Schema) $E_jU \Rightarrow_s I^*_s OE_jC$

In FLBC we have

Expression 8 (FLBC Commit Schema) $\mathit{commit}(e_1) \wedge \mathit{Speaker}(e_1, j) \wedge \mathit{Addressee}(e_1, k) \wedge \mathit{Cul}(e_1, t_1) \wedge \mathit{Content}(e_1, \lceil C \rceil)$

along with

Axiom Schema 7 (FLBC Commit Axiom Schema) $(\mathit{commit}(e) \wedge \mathit{Content}(e, \lceil C \rceil)) \rightarrow (\mathit{Kept}(e) \leftrightarrow C)$

In FLBC, promises (and commissives generally) are characterized by whether they are kept or not. Jones's analysis undertakes to represent the deontic consequences of making a promise and in general of giving a commitment.

The usual pattern applies when the two perspectives are integrated.

Axiom Schema 8 (Governing Commissive Axiom Schema) $E_j(\mathit{commit}(e) \wedge \mathit{Speaker}(e, j) \wedge \mathit{Addressee}(e, k) \wedge \mathit{Cul}(e, t) \wedge \mathit{Content}(e, \lceil C \rceil)) \Rightarrow_f I^*_f OE_jC$

Points arising:

1. As noted above, there may be need for a more resolved view of obligation. Different institutions may require different commitments, so obligations may need to be relativized to institutions.
2. For some purposes a more detailed modeling may be appropriate. For example, promises are usually seen as applying only to the future. A promise to do something in the past is infelicitous. Such constraints can be added to the formalism we introduce. We leave the details for future work. In many practical situations, however, these kinds of refinements may not be necessary.

3.3 Declaratives

A declarative signalling act (in ideal or felicitous conditions) brings about the state of affairs described by its content. Sometimes saying so indeed makes it so. A state of war may be brought about by declaring it to be the case. Similarly, naming (by parents or other authorities) and pronouncing (by juries, judges, clerics, and referees) are acts that will typically bring about the conditions described by their contents.

Jones's analysis, and the corresponding FLBC representations, follow the usual patterns.

Expression 9 (Jones's Declarative Schema) $E_jU \Rightarrow_s I_s^* E_j C$

Under Jones's analysis j's bringing about the utterance U counts, under ideal circumstances, as j's bringing about the state of affairs described by C. In FLBC we have

Expression 10 (FLBC Declaration Schema) $\mathit{declare}(e_1) \wedge \mathit{Speaker}(e_1, j) \wedge \mathit{Addressee}(e_1, k) \wedge \mathit{Cul}(e_1, t_1) \wedge \mathit{Content}(e_1, \lceil C \rceil)$

along with

Axiom Schema 9 (FLBC Declaration Axiom Schema) $(\mathit{declare}(e) \wedge \mathit{Content}(e, \lceil C \rceil)) \rightarrow (\mathit{Authoritative}(e) \rightarrow C)$

Again, the usual pattern applies when the two perspectives are integrated.

Axiom Schema 10 (Governing Declarative Axiom Schema) $E_j(\mathit{declare}(e) \wedge \mathit{Speaker}(e, j) \wedge \mathit{Addressee}(e, k) \wedge \mathit{Cul}(e, t) \wedge \mathit{Content}(e, \lceil C \rceil)) \Rightarrow_f I_f^* E_j C$

Regarding the FLBC axiom schema, the special predicate *Authoritative* must be interpreted carefully. In a sporting event, such as baseball, an umpire may declare a player out. Which umpire has the authority to do so depends upon the state of play, and so an umpire may say a player is out but lack the proper authority and in fact be overruled by another umpire. But there is nothing in principle to prevent two umpires, one lacking authority, *both* to declare a player out. Consequently, saying without authority that a player is out is *not* sufficient for the player to be safe (not out). The usual biconditional has to be replaced by the weaker conditional.[5]

[5] In fact each of the FLBC speech act auxiliaries—*Kept, Honored, Authoritative, Veridical*—merit careful discussion and interpretation. That, however, is beyond the scope of this paper. We note, however, that the event indices in FLBC, e_1, e_2, etc., may be exploited to mark the cause of a truth. C may be true *by virtue of* e_2 but not true by virtue of e_3.

4 Discussion: Towards Deployment

The logical analysis of signalling acts will, we hope, contribute to electronic commerce in each of the three ways identified in the Introduction: as a means of clarification, as a guide to design, and as a basis for implementation. To that end we offer the following remarks, more to spur discussion than to complete it.

4.1 Clarification: Intentions and Speech Acts

Speech act theory is not without contending schools of thought. Here we wish to comment briefly on the debate between the intentionist and conventionist views regarding the proper analysis of speech acts. Despite his criticisms of the Gricean intention-based theory of meaning [Gri57], Searle will for present purposes be taken to be one of the many representatives of the intentionist theory.[6] See also many of the papers in [CMP90] and much of the computer science literature, e.g., [Sin93,SC96], plus KQML[7] and other agent communication languages. The following passage from Searle is representative of intentionist theory for speech acts as we shall discuss it here.

> So far we have considered only the case of a sincere promise. But insincere promises are promises nonetheless, and we now need to show how to modify the conditions to allow for them. In making an insincere promise the speaker does not have all the intentions he has when making a sincere promise; in particular he lacks the intention to perform the act promised. However, he purports to have that intention. Indeed, it is because he purports to have intentions which he does not have that we describe his act as insincere.
> A promise involves an expression of intention, whether sincere or insincere. So to allow for insincere promises, we need only to revise our conditions to state that the speaker takes responsibility for having the intention rather than stating that he actually has it. A clue that the speaker does take such responsibility is the fact that he could not say without absurdity, e.g., "I promise to do A but I do not intend to do A". To say, "I promise to do A" is to take responsibility for intending to do A, and this condition holds whether the utterance was sincere or insincere. To allow for the possibility of an insincere promise, then we have only to revise condition 6 so that it states not that the speaker intends to do A, but that he takes responsibility for intending to do A, ... [Sea69, page 62]

[6] Bach and Harnish [BH79] adopt a strong intentionist view of speech acts and communication. They see Searle as holding a conventionist view in contrast to their intention-and-inference theory. These are issues beyond the scope of this paper.

[7] E.g., [Cov98], [FFMM94a], [FFMM94b], [FW$^+$93], [LF94], [MLF96], and [Moo00a]. See for KQML: http://www.cs.umbc.edu/kqml/.

Searle's claim that "it is because he purports to have intentions which he does not have that we describe his act as insincere" is not an argument for his intentionist position so much as a restatement of it. Falsely purporting is of course a reason for attributing insincerity. So, according to a conventionist, would be invoking a convention without intending to meet the obligations that come with it. By performing a signalling act of the commissive type, the agent places himself under an obligation (to do A). There is nothing *absurd* in his saying, additionally, that he does not intend to do A, although it might be unwise for him to explicitly reveal his insincerity in this way. Insincerity can be incorporated without recourse to a change in the way promising itself is understood. In particular, there is no need to resort to a revised condition of the type proposed by Searle—unless of course we are to understand 'taking responsibility for intending to do A' as merely a roundabout way of saying 'placing oneself under an obligation to do A'.

The core of the conventionist view is that communicative acts occur in the context of governing conventions. These conventions stipulate the states of affairs that should obtain in consequence of performance of the communicative acts themselves. Under Jones's analysis of the convention of committing, Expression 7, if j performs a signalling act that counts in s as a commissive (with respect to C), then j is placed under an obligation to see to it that C, unless certain ideality conditions are not met. What conditions are these? Well, in particular, j will have to be an agent who is empowered to place himself under an obligation. Among human agents, a minor, for instance, might not be so empowered. Among electronic agents, the institutional arrangement might be that only certain categories of agent are so empowered.

To this the intentionist might perhaps respond: whether or not j's performance of the given signalling act does indeed count as a commissive will depend on the intentions j has when he performs the act. For suppose he sends the signal by accident, or is play-acting, or just joking, would we then say that he has placed himself under an obligation? But here we are back to the issue addressed above in point 4c, page 329, pertaining to Axiom Schema 2. If the communicating agent, or its owner, wishes to maintain that the communicative act was not *serious*, not a *literal* implementation of the governing convention, then the onus of proof will be on the sender or owner to show just why it was not.

4.2 Design and Implementation

Kimbrough's FLBC was designed with an eye to implementation. The formulation in first-order logic, even with quotation, lends itself well to logic programming formalisms. This facilitates implementation either in a logic programming environment or in more conventional programming system.[8] Combining FLBC with Jones's analysis of signalling conventions, however,

[8] See the work of Alan Abrahams in this regard, e.g., [Abr02,AEB04].

introduces several new sentence operators, each with its own logic, notably I^*, E_j, and O. We shall now present an example that illustrates how our integration of FLBC and Jones's analysis of signalling conventions might be implemented in a logic programming formalism. Our purpose is merely to sketch a plausibility case. The full case requires a lengthy and detailed treatment, which is beyond the scope of this, or any single, paper.

Consider as an example of a content sentence, C:

Expression 11 *Andrea Doria leaves Boston on January 1, 2005.*

Here is an FLBC representation:

Expression 12 *leave*(*e1*) ∧ *Experiencer*(*e1, Andrea Doria*) ∧ *Goal*(*e1, Boston*) ∧ *Cul*(*e1, 20050101*)

Note that *e1* is (in virtue of the covering interchange agreement) a unique ID for Expression 12. Also, *Cul* is a special predicate indicating the time of an event, here *e1*. Converting Expression 12 to a Prolog representation in which the expression is the quoted part of a *Content* predicate gives us Expression 13.

Expression 13
```
content(e2, (leave(e1),
experiencer(e1, 'Andrea Doria'),
goal(e1, 'Boston'), cul(e1, '20050101'))).
```

Suppose now that Bob promises Carol that Andrea Doria leaves Boston on January 1, 2005. In FLBC:

Expression 14 *promise*(*e2*) ∧ *Speaker*(*e2, 'Bob'*) ∧ *Addressee*(*e1, 'Carol'*) ∧ *Content*(*e2,*⌈*leave*(*e1*) ∧ *Experiencer*(*e1, 'Andrea Doria'*) ∧ *Goal*(*e1, Boston*) ∧ *Cul*(*e1, 20050101*)⌉) ∧ *Cul*(*e2, 20041201*)

In Prolog, a message/2 clause:

Expression 15 (Utterance Example in Prolog)
```
message(e2, (promise(e2),
speaker(e2, 'Bob'), addressee(e2, 'Carol'),
content(e2, (leave(e1) , experiencer(e1,
'Andrea Doria') , goal(e1, 'Boston'),
cul(e1, 20050101))) , cul(e2, 20041201))).
```

At this point we make contact with Jones's analysis of signalling conventions. We need a Prolog representation of

Axiom Schema 11 (Governing Promising Axiom Schema)
$E_j(promise(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Cul(e, t) \wedge Content(e, \lceil C \rceil)) \Rightarrow_f I_f^* O E_j C$

Our representation will be approximate, and in two parts. First, $\Rightarrow_f$, the 'counts as' connective in Axiom Schema 11, also the main connective in the expression, will be represented by a Prolog predicate of arity 2, `countsas`:

Expression 16 (Promising Axiom Schema in Prolog)

```
countsas(istar(obligation(
(stit(E),agent(E,S),content(E,C)))),
message(E,(promise(E),speaker(E,S),
addressee(E,A),content(E,C),cul(E,T)))).
```

Note that the first argument,

```
istar(obligation(stit(E,C))),
```

corresponds to the right-hand side of Axiom Schema 11, and the second argument is the schema for a `message` (see Expression 15). `E`, `A`, `C` and `T` are free variables (subject to uniform substitution), permitting Expression 16 to act as a general rule, or axiom schema. If we are to deduce much from Expression 16 we need a second part for our representation. There are a number of ways to achieve this, but perhaps the simplest is to add a general rule that supports a form of detachment based on Expression 16. We can do this in Prolog with Expression 17.

Expression 17 (Promising Rule Schema in Prolog)

```
istar(obligation(
(stit(E),agent(E,S),content(E,C)))) :-
countsas(istar(
obligation((stit(E),agent(E,S),
content(E,C)))),
message(E,(promise(E),speaker(E,S),
addressee(E,A), content(E,C),cul(E,T)))),
message(E,(promise(E),speaker(E,S),
addressee(E,A),content(E,C),cul(E,T))).
```

Taken together, Expressions 15, 16 and 17 support a deduction in Prolog that, in ideal circumstances, there is an obligation that Bob sees to it that there is a leaving by Andrea Doria for Boston on January 1, 2005.

```
?- istar(X).

X = obligation((stit(e2), agent(e2, 'Bob'),
content(e2, (leave(e1),
experiencer(e1, 'Andrea Doria'),
goal(e1, 'Boston'),
cul(e1, 20050101)))))
```

Adding new messages having the form of Expression 15 will allow additional such deductions to be made.

The illustration just given of an implementation in Prolog is very much an approximation and obviously requires much to be completed. Even so, it suits our present purpose, which is to further the plausibility of a thoroughgoing logicist approach to the Basic Messaging Framework, Figure 1. In furtherance of that end, we conclude this section with a discussion of how the *stit* (sees to it that) operator, E_j, can be articulated in FLBC. The Prolog exercise just concluded should be sufficient to show that a subsequent translation from FLBC to logic programming may be made straightforwardly. We acknowledge this as a promissory note and add to the list detailed articulation of the countsas operator ($\Rightarrow_s$), the I-star operator (I_s^*), and the deontic operators, including that for obligation O.

4.3 *stit* in FLBC

From action logic, E_jA, or 'j sees to it that A', is rendered into FLBC as:

Expression 18 (FLBC Stit Schema) $stit(e_1) \wedge Agent(e_1, j) \wedge Cul(e_1, t_1) \wedge Content(e_1, \lceil A \rceil)$

Action logic introduces a second operator, C_j, called *capacitation.* C_jA may be interpreted as 'j has the capacity (or ability) to produce the state of affairs described by A'. This is represented in FLBC as:

Expression 19 (FLBC Capacitation Schema) $capacitation(e_1) \wedge Agent(e_1, j) \wedge Hold(e_1, t_1) \wedge Content(e_1, \lceil A \rceil)$

(Rough translation: e_1 is a particular capacitation state; j is an agent in that state; e_1 has content $\lceil A \rceil$, and the state obtains, or holds, at time t_1. Note that events are said to happen or culminate (*Cul*) at a given time, and states are said to obtain or hold (*Hold*) at a given time. See Parsons [Par90] for exposition on this distinction.)

We assume a standard version of action logic, as presented in [Jon04], which is characterized by a series of axiom schemas or rules of inference. We need FLBC analogs to them. In the original logic seven principles apply. The first is:

$$\text{If} \vdash A \leftrightarrow B \text{ then} \vdash E_jA \leftrightarrow E_jB \tag{1}$$

Translation: If A and B are logically equivalent, infer that E_jA and E_jB are logically equivalent. The analog for *stit* is

Expression 20 If $\vdash A \leftrightarrow B$ then
$\vdash stit(e) \wedge Agent(e, i) \wedge Content(e, \lceil A \rceil) \leftrightarrow$
$stit(e) \wedge Agent(e, i) \wedge Content(e, \lceil B \rceil)$

A corresponding rule of inference applies to C_j:

$$\text{If} \vdash A \leftrightarrow B \text{ then} \vdash C_jA \leftrightarrow C_jB \tag{2}$$

Its FLBC translation is:

Expression 21 If $\vdash A \leftrightarrow B$ then $\vdash$ *capacitation*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil) \leftrightarrow$ *capacitation*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil B \rceil)$

The five remaining principles are axiom schemas in the original action logic.

$$E_i A \rightarrow A \tag{3}$$

In FLBC:

Expression 22 *stit*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil) \rightarrow A$

$$(E_i A \wedge E_i B) \rightarrow E_i (A \wedge B) \tag{4}$$

In FLBC:

Expression 23 (*stit*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil) \wedge$ *stit*$(e') \wedge$ *Agent*$(e', i) \wedge$ *Content*$(e', \lceil B \rceil)) \rightarrow$ *stit*$([e, e']) \wedge$ *Agent*$([e, e'], i) \wedge$ *Content*$([e, e'], \lceil A \wedge B \rceil)$

$$\neg E_i \top \tag{5}$$

Similarly for C_i:

$$\neg C_i \top \tag{6}$$

In FLBC we add rules of inference:

Expression 24 (E_i Rule) If $\vdash A \leftrightarrow \top$ then $\vdash \neg$(*stit*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil))$

Expression 25 (C_i Rule) If $\vdash A \leftrightarrow \top$ then $\vdash \neg$(*capacitation*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil))$

Finally:

$$E_i A \rightarrow C_i A \tag{7}$$

In FLBC:

Expression 26 *stit*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil) \rightarrow$ *capacitation*$(e) \wedge$ *Agent*$(e, i) \wedge$ *Content*$(e, \lceil A \rceil)$

5 Summary and Conclusion

The Basic Messaging Framework, Figure 1, describes the communication setup for electronic commerce. In that framework the Interchange Agreement plays a large and essential role. The Interchange Agreement is, at bottom, a conceptual abstraction that covers the conventions needed to conduct business transactions. Consequently, seeing speech acts as events in a conventional signalling system fits naturally with the Basic Messaging Framework: the underlying speech act conventions are simply part of the Interchange

Agreement. Formalizing any of the conventions in the Interchange Agreement has the salutary prospect of clarifying concepts, of guiding design, and even of supporting implementation.

We have essayed in this paper a demonstration of how speech act conventions as formalized by Jones's work may be made to fit with Kimbrough's FLBC formalization of messages. The project is far from complete. There are many technical and conceptual alternatives that merit investigation, and scaled-up implementation will be required to test fully these ideas. Our goal here has been the modest one of presenting necessary components for a thorough formal modeling of communicative acts in the context of electronic commerce. Speech acts are among the most important of these components, as is their attending inferential apparatus. A philosophically sound formalization of speech acts has been unified with an expressively powerful message representation formalism, and a sketch has been made of how the combined result could be rather directly representable in Prolog. Very much remains to be done to redeem the promissory notes issued. Even so, this is, we submit, a strong and favorable indicator for the near-term prospects of a strong logicism in electronic commerce.

References

[Abr02] Alan S. Abrahams, *Developing and executing electronic commerce applications with occurrences*, Ph.D. thesis, University of Cambridge Computer Laboratory, 2002.

[AEB04] Alan S. Abrahams, David M. Eyers, and Jean M. Bacon, *Practical contract storage, checking, and enforcement for business process automation*, Formal Modelling for Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, Berlin, Germany, 2004.

[Aus62] John L. Austin, *How to do things with words*, Oxford at the Clarendon Press, Oxford, England, 1962.

[BH79] Kent Bach and Robert M. Harnish, *Linguistic communication and speech acts*, The MIT Press, Cambridge, Massachusetts, 1979.

[CMP90] Philip R. Cohen, Jerry Morgan, and Martha E. Pollack (eds.), *Intentions in communication*, System Development Foundation Benchmark, The MIT Press, Cambridge, Massachusetts, 1990.

[Cov98] Michael A. Covington, *Speech acts, electronic commerce, and KQML*, Decision Support Systems **22** (1998), no. 3, 203–211.

[FFMM94a] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire, *KQML as an agent communication language*, Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94), ACM Press, November 1994.

[FFMM94b] ———, *KQML as an agent communication language*, The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94), ACM, ACM Press, November 1994.

[FW+93] Tim Finin, Jay Weber, et al., *Draft specification of the KQML agent–communication language plus example agent policies and architectures*, Manuscript obtained from `http://www.cs.umbc.edu`, 1993.

[Gri57] Paul Grice, *Studies in the way of words*, ch. Meaning, pp. 213–223, Harvard University Press, Cambridge, MA, 1989 (originally 1957), .

[HS98] Michael N. Huhns and Munindar P. Singh (eds.), *Readings in agents*, Morgan Kaufmann, San Francisco, CA, 1998, ISBN: 1-55860-495-2.

[Jon02] Andrew J.I. Jones, *On the concept of trust*, Decision Support Systems **33** (2002), no. 3, 225–232.

[Jon04] ———, *A logical framework*, Open Agent Societies: Normative Specifications in Multi-Agent Systems (Jeremy Pitt, ed.), John Wiley & Sons, Chichester, UK, 30 December 2004, ISBN: 047148668X.

[JP04] Andrew J.I. Jones and X. Parent, *Conventional signalling acts and conversation*, Advances in Agent Communication (Frank Dignum, ed.), Lecture Notes in Artificial Intelligence, Volumn 2922, Springer-Verlag, Berlin, Heidelberg, New York, 2004, pp. 1–17.

[JS96] Andrew J.I. Jones and Marek J. Sergot, *A formal characterisation of institutionalised power*, Journal of the Interest Group in Pure and Applied Logic (IGPL) **4** (1996), no. 3, 427–443, Reprinted in [V+97, pages 349–367].

[Kim99] Steven O. Kimbrough, *Formal language for business communication: Sketch of a basic theory*, International Journal of Electronic Commerce **3** (Winter 1998–99), no. 2, 23–44.

[Kim01] ———, *Reasoning about the objects of attitudes and operators: Towards a disquotation theory for representation of propositional content*, Proceedings of ICAIL '01, International Conference on Artificial Intelligence and Law, 2001.

[Kim02] ———, *A note on the Good Samaritan paradox and the disquotation theory of propositional content*, Proceedings of ΔEON'02, Sixth International Workshop on Deontic Logic in Computer Science (John Horty and Andrew J.I. Jones, eds.), May 2002, pp. 139–148.

[KM97] Steven O. Kimbrough and Scott A. Moore, *On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity*, ACM Transactions on Information Systems **15** (October 1997), no. 4, 321–367.

[KT00] Steven O. Kimbrough and Yao-Hua Tan, *On lean messaging with unfolding and unwrapping for electronic commerce*, International Journal of Electronic Commerce **5** (2000), no. 1, 83–108.

[LF94] Yannis Labrou and Tim Finin, *A semantics approach for KQML—a general purpose communication language for software agents*, Third International Conference on Information and Knowledge Management (CIKM '94), November 1994.

[MLF96] James Mayfield, Yannis Labrou, and Tim Finin, *Evaluation of KQML as an agent communication language*, Intelligent Agents Volume II – Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (Berlin, Germany) (M. Wooldridge, J. P. Muller, and M. Tambe, eds.), Springer-Verlag, 1996.

[Moo00] Scott A. Moore, *KQML and FLBC: Contrasting agent communication languages*, International Journal of Electronic Commerce **5** (2000), no. 1, 109–124.

[Par90] Terence Parsons, *Events in the semantics of English: A study in subatomic semantics*, Current Studies in Linguistics, The MIT Press, Cambridge, MA, 1990, ISBN: 0-262-66093-8.

[SC96] Ira A. Smith and Philip R. Cohen, *Toward a semantics for an agent communication language based on speech acts*, Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vol. 2 (Menlo Park, California) (Howard Shrobe and Ted Senator, eds.), AAAI Press, 1996, pp. 24–31.

[Sea69] John R. Searle, *Speech acts*, Cambridge University Press, Cambridge, England, 1969.

[Sin93] Munidar P. Singh, *A semantics for speech acts*, Annals of Mathematics and Artificial Intelligence **8** (1993), no. I–II, 47–71, Reprinted in [HS98].

[V+97] E. Garzòn Valdés et al. (eds.), *Normative systems in legal and moral theory – festschrift for Carlos E. Alchourrón and Eugenio Bulygin*, Duncker & Humblot, Berlin, Germany, 1997.

Dynamic Conversation Structures: An Extended Example

Scott A. Moore

University of Michigan Business School, Ann Arbor, MI, USA,
samoore@umich.edu

Abstract. The author provides an in-depth look at a moderately complex conversation as represented by a finite state machine, a representation used by an established agent communication system. He compares this to a statechart-based method used for Moore's conversation policy framework and observes that these methods differ in their level of detail, the usefulness of parts of the graphical representation, and in the grouping of events. The remainder of the paper demonstrates how a multi-agent conversation policy can be used to control the flow of messages, contrasts this with how messages are handled via an inference-based process, and shows how the inference-based processing can be integrated with the policy-based handling in order to deal with exceptions to the policy.

1 Introduction

Researchers have proposed many agent communication languages (e.g., FIPA's ACL [Fou97], FLBC [Moo01], KQML [LF97]). In the development of each of these languages, it has become apparent that there needs to be a method for modeling conversation policies (see the collection in [DG00]). Moore defined a method of representing conversation policies, applied it to FLBC, and demonstrated it by modeling a series of relatively simple conversations [Moo00b]. In this paper I apply this method to a more complex conversation. This example shows how this method works and how the system can be used to coordinate the actions of two conversational participants.

A conversation among agents is composed of an exchange of messages. This exchange is generally aims at the accomplishment of some task or the achievement of some goal. In its simplest form it is a sequence of messages in which, after the initial message, each is a direct response to the previous one. More complicated structures occur when subdialogs are needed. Linguists and philosophers have not come to any consensus about subdialogs, but several types consistently appear:[1] subordinations, corrections, and interruptions. A message begins a *subordinate* conversation when it elaborates on a point made in a previous message. This message should be about the previous message, probably as a clarification of some fact. A message that begins a *correction*

[1] See [LA87,Moo01,Pol88].

subdialog indicates that the message somehow corrects a previous message in the conversation. A message *interrupts* the current conversation when it is neither an expected nor the standard reply to the previous message. This should be used only if the other two do not apply.

A conversation policy (CP) defines 1) how one or more conversation partners respond to messages they receive, 2) what messages one partner expects in response to a message it sends, and 3) the rules for choosing among competing courses of action. These policies can be used both to describe how a partner will respond to a message (or series of messages) it receives and to specify the procedure the partner actually executes in response to that message (or those messages). The policy is a template describing an agent's reactions to an incoming message, its expectations about upcoming messages, and the rules it applies to determine its own reactions. In a linguistic sense, it moves the conversation from an inference-based process to a convention-based one. Inference acts as a backup, providing a more computationally expensive way of understanding unfamiliar messages for which CPs have not been defined. The means by which this backup can be implemented is demonstrated in §2.

Conversation policies are represented using the statechart formalism defined by Harel [Har87] and then later integrated into the UML.[2] This is a graphical language which developers should find easier to work with than, e.g., an underlying XML representation which the agents themselves might use. Statecharts are quite appropriate for event-driven systems, conveniently representing "the set of allowed sequences of input and output events, conditions, and actions, perhaps with some additional information such as timing constraints." [Har87, pp. 231–2].

Since CPs are used to govern an agent's behavior in a conversation, each one begins with a message being either sent or received. The sent or received message acts as a tag for the CP, indicating when it will be invoked. An agent maintains two databases related to CPs. One is the set of CPs it can use and the other is the set of currently executing CPs. Figure 1 contains a sample statechart representation of a conversation policy. The whole state-

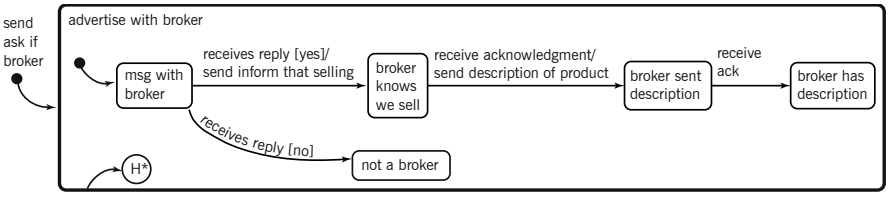


Fig. 1. A sample statechart representation of a conversation policy

[2] [BRJ99,RJB99]

chart is labeled advertise with broker. It has six states: advertise with broker, msg with broker, broker knows we sell, broker sent description, not a broker, and broker has description. Transition labels have the form t[c]/g, each part of which is optional. When the event represented within the trigger t occurs, then, if the condition c is met, the transition between states is made. If the g is present on the transition, then the event g is generated as the transition is taken.

2 Extended Example

In this section I explore in detail an example adapted from Greaves, et al. [GHB99]. Their example is specified for the KAoS system [BDBW97] using a form of finite state machine (FSM); this is shown in Figure 2. I will first go through an analysis of the representation used by Greaves et al. and compare it with the CP-based representation. I then show how the CP can be used as a control for an agent's basic processing of messages; I follow this by showing how CPs can be integrated into a system that uses inference-based messsage handling.

2.1 Analysis of Representations

Here is an interpretation of this finite state machine:

> The contractor announces a request for bids. At this point the contractor can withdraw, the supplier can fail to respond to the request, or the supplier can submit a bid. After the bid is received by the contractor, the contractor can either reject the bid, withdraw the original offer, fail to respond to the bid, or send a message of acceptance.
>
> After the message of acceptance is sent, either the supplier or contractor can withdraw or renege. If neither of these happen, then the task is complete.

Figure 3 shows a representation of this basic process as a conversation policy (and as a statechart). The sequencing and policies covering the conversation are essentially equivalent to the specification shown in Figure 2 though there are many differences in detail. The actions of the two conversation participants are separate in the current version while the FSM-based version integrate their actions. The top half of this figure describes the contracting process for the contractor; the bottom half is for the supplier.

In order to understand this CP, we need to examine it in context of all of the CPs governing the agent's behavior. For this, consider Figure 4. Each agent has one control process and many CPs. In this diagram only one policy is shown. For each CP one transition to it away from the control process, and it is labelled with the message that the agent sends. The sending of this message causes control to transfer from the control state to the CP. When

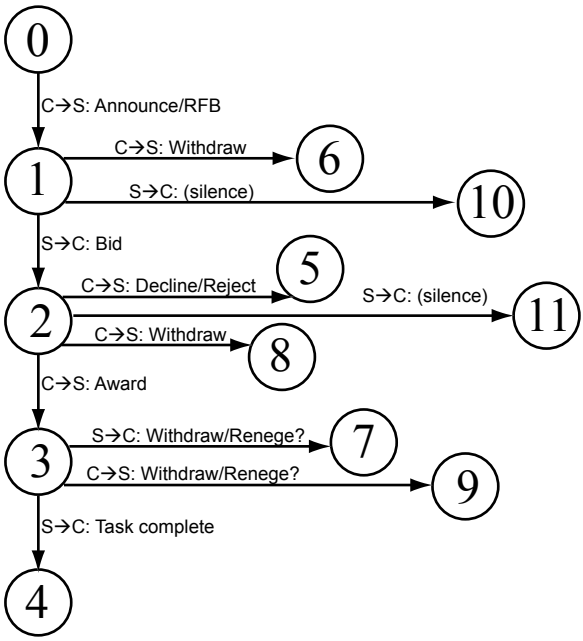


Fig. 2. 'Contract' conversation policy as represented by a finite state machine for the KAos system [BDBW97]. Note: The transition from 2 to 11 is labelled as S→C: (silence). Since every other transition from state 2 involves the contractor sending a message to the supplier, I believe that this should be C→S: (silence) (i.e., the lack of a the contractor sending a message). For the rest of this document I assume that this correction has been made.

the CP is done executing, the system takes the one transition back from the CP to the control state. Thus, when the contractor sends the request for bid message, the contract process state is entered. Figure 4 shows the system's overall structure while Figure 3 shows the internal structure of the contract process state.

The following describes the correspondance between the FSM and CP versions. At the beginning of each discussion point is a pair **x-y**; this refers to the connection between states **x** and **y** in Figure 2. The list of numbers immediately following this pair refers to the numbered transitions in Figure 3.

0-1 : 1, 10. This transition represents the contractor sending out a request for bid message. Control has passed to the contract process state so we can assume that a request for bid message has already been sent, thus passing control to this CP. Once this state is activated, then the contractor follows the default transition (labelled transition 1 in Figure 3).
The contractor sends the request to the supplier who receives the request and, thus, follows transition 10. In this case one transition in the FSM

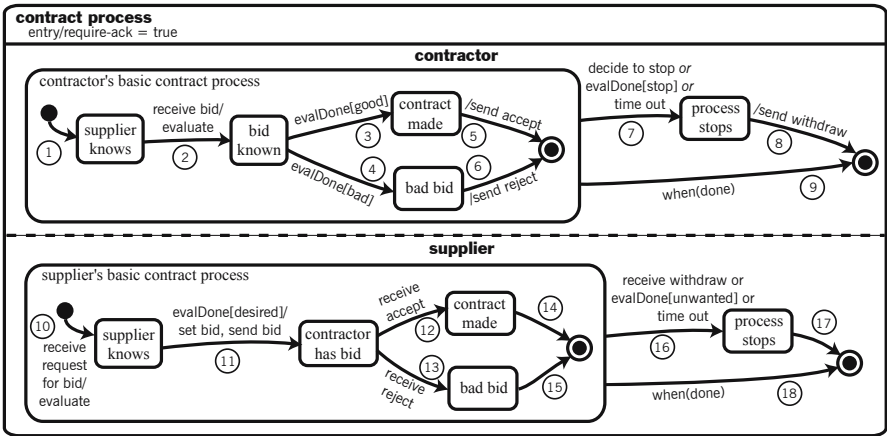


Fig. 3. Conversation policy for the control process. This is invoked in response to a 'send request for bid' message. Note: The circled numbers are not part of the conversation policy; they are included for purposes of exposition.

Fig. 4. Controlling process for conversation policies

version corresponds to two transitions in the CP because the CP models the actions of the conversation partners at a lower level of detail. Here, the sending of the message does not imply the receiving of the same message; this would be appropriate in any electronic community that either is based on technology with less than 100% reliability or uses technology for sending messages in which the sending and receiving of messages occur at different times.

1-6 : 7, 8, 16, 17. The FSM version simply states that the contractor sends a withdraw message to the supplier. The CP version has more detail. In this version for the process to stop the contractor either decides to stop, finishes some evaluation process that indicates that it should stop, or the process simply times out (from the inactivity of either partner). (The semantics of statecharts is such that the transition from a superstate is inherited by all of its substates; thus, if the process is in any of the substates within contractor's basic contract process, then the process will exit the substate when an event occurs that triggers a transition from the superstate.) In this case the contractor decides to stop the process for

some reason other than the bid's quality (since it has not even received the bid yet).

When the contractor decides to stop the process, the process follows transition 7 and then immediately follows transition 8 and, in doing so, sends a withdraw message to the contractor. The receiving of the withdraw message causes the supplier to follow transition 16 and then transition 17. Since both halves of the statechart have reached the final states, the whole statechart is completed.

1-10 : 7, 8, 16, 17. This step indicates that the supplier does not send a bid in response to the request. The lack of action by the supplier, resulting in a time out event to occur, causes the supplier's process to take transitions 16 and 17. Eventually this lack of action will cause a time out in the process for the contractor as well, which triggers it to take transitions 7 and then 8. The result is that the contractor sends a withdraw to the supplier. Sending the message differs from the FSM version but seems consistent with the scenario. If this were not desired, then the CP could be modified so that a separate transition were defined for time out that did not result in the sending of a message.

1-2 : 2, 11. In this step the supplier sends a bid back to the contractor. The CP again separates this into separate actions. The supplier has finished the evaluation that it began in transition 10 and the result is that it desires to send a bid (see evalDone(desired) on transition 11). After this event occurs, the supplier sets its bid and then sends the bid back to the contractor (again, transition 11). When the contractor receives the bid, it then begins evaluating the bid (transition 2).

2-5 : 4, 6, 9, 13, 15, 18. This step indicates that the contractor has decided to reject the supplier's bid. When the contractor finishes its evaluation and determines that the bid is bad, then it takes transition 4 into the bad bid state and then immediately follows transition 6 and sends a reject message. After sending this message the process then follows transition 9 since the contractor's basic contract process is done. When the supplier receives the reject message, it takes transition 13 into the bad bid state and then immediately takes transition 15 and then transition 18. Further, since both substates have completed, then this process is done and control goes back to the control state in Figure 4.

2-8, 2-11, 3-7, 3-9 : 7, 8, 16, 17. These steps indicate other possible times for the contractor to withdraw its request for bid. On the statechart this is again represented by the same transitions as was used for **1-10**. On the FSM the termination of the bidding process for reasons other than simply rejecting the bid is represented by states 6, 10, 11, 8, 7, and 9; on the statechart version it is represented by transition 7 into the process stops state for the contractor and transition 16 for the supplier.

2-3 : 3, 5, 9, 12, 14, 18. This represents the awarding of the contract, that is, the sending of an acceptance message from the contractor to the supplier. On the statechart this is represented by the generation of an event

signalling that the evaluation is done and the determination that the bid is good (transition 3 into the contract made state). This is immediately followed by the contractor sending an acceptance message, following transition 5 into the final state of the contractor's basic contract process, and then following transition 9 into the final state of the contract process. When the supplier receives the accept message, it follows transition 12 into the contract made state. It then immediately follows transitions 14 and 18. As above, control is now passed back to the overall control state.

3-4 : It is unclear what this transition does. In any case it does not appear to be anything that affects the current analysis.

Given the above analysis we can make some observations about the the relationship between these two representations related to their level of detail, the usefulness of state names, and the grouping of events.

Level of detail The CP approach shows the process at a greater level of detail than the FSM approach. This might be merely indicative of the example Greaves et al. chose or it might accurately represent the purpose for which the FSM is used — it is unclear. The CP approach aims to identify and model the activities related to interaction with other agents plus any other actions that affect the timing or contents of any messages that flow among conversational partners. Further, the activities of each participant in the conversation must be shown separated from the others.

Usefulness of state names Regardless of the level of detail shown in the models, the statechart representation is more explanatory by its nature than the FSM example shown. For example, remove the labels on all of the transitions in the CP and it is still quite clear as to what is going on in the CP. Adding the labels back in provides more insight into the process. The names of the states are meant to help describe the current state of the system; clearly, the more descriptive and accurate this description is, the more it is seen as an advantage over the numbered states of the FSM approach.

Grouping of events The CP approach groups the events by user. This has obvious benefits and drawbacks — sometimes an analyst wants to see events grouped by user, and sometimes he wants to see how the events relate to one another. Another point related to grouping is that the CP tends to group all interruptions to the standard process (e.g., that contained in contractor's basic contract process) so that the regular flow of control is more apparent. When looking at the FSM-based representation, it is perhaps not as clear as it could be that states 6, 10, 11, 8, 7, and 9 are all, more or less, equivalent states; the CP representation makes this more clear.

2.2 Basic Processing

We have now presented an analysis in detail of these two methods of representing conversations. The purpose of the following exposition is to demonstrate how a multi-agent conversation policy can be used to control the flow of messages, contrast this with how messages are handled via an inference-based process, and show how the inference-based processing can be integrated with the policy-based handling in order to deal with exceptions to the policy.

In the following I list the messages that might be sent in fulfillment of this conversation policy. I use a `Prolog`-based representation for the FLBC messages for space considerations. The `send(A, B, C)` predicate should be interpreted as Agent `A` sends message Agent `C` to the agents listed in `B`. Further, `msg(S, R, F, C, X)` should be interpreted as speaker `S` sending a message to recipient `R` with illocutionary force `F`, content `C`, and with relevant context `X`.

The message in Figure 5 is sent from Agent `C` to Agent `S1` (line #1). This

```
send(c, [s1],
  msg(c, s1, request,
      send(s1, c,
        msg(s1, c, offer,
         [reserve(z),
          Agent(z, s1),
          Object(z, x),
          room(x),
          beginDate(z, time(1999, 6, 6)),
          endDate(z, time(1999, 6, 10)),
          location(x, delft)])),
     [cp(contractProcess),
      convID(v423),
      ackPolicy([c, s1], [parse], yes),
      timeSent(time(1999, 5, 11)),
      msgID(c45)]))
```

Fig. 5. Message #1

is a request from the sender to the recipient that the recipient send back to Agent `C` an offer in which Agent `S1` reserves a hotel room for Agent `C` in Delft from June 6 to June 10. Of course, predicates in this term are simplified versions of what actual applications would require; for instance, information about the room and the location would have to be markedly more detailed.

When this message is sent by Agent `C`, the "contract process" conversation policy is invoked since both the `cp()` term in the context matches the process's name and the message itself matches the process's trigger (see Figure 4). The default transition in the top half is taken to put the contractor in the "supplier knows" state.

When the supplier receives this message, it determines from the `ackPolicy()` term that it must acknowledge to the contractor when it successfully parses this message. This requirement of "parsing" is more restrictive than simply receiving the message but is less so than fully processing the message. This acknowledgment should indicate to Agent C that the sender has been able to parse the message and knows what each of the terms mean. This acknowledgment does *not* tell Agent C either that the sender has evaluated the original message or that the sender is going to respond to the message. The purpose of this message is to confirm that the receiving agent is on-line and that the agent communication system is successfully routing messages. Agent S1 sends Message #2 back to Agent C (see Figure 6). Having

```
send(s1, c,
  msg(s1, c, inform,
      parse(s1, msgID(c45)),
      [cp(contractProcess),
       convID(v423),
       msgID(s56)]))
```

Fig. 6. Message #2

received this acknowledgment, Agent C can reasonably assume that Agent S1 received the message. This takes one level of uncertainty out of the processes of interpreting messages and managing conversations.

For the most part these acknowledgment messages lie outside the part of the message processing framework that I want to concentrate on. Notice, however, that the sequencing of the contract process conversation policy remains the same whether or not an acknowledgment is requested. The decision about acknowledgment could be defined as a variable to be agreed upon by the conversants before the conversation begins. It could even be agreed upon for all conversations between the two parties when they first begin to exchange messages. In fact Figure 3 shows an example of this in the upper-left corner just under the name of the CP: the phrase `entry/ require-ack = true`. This is how UML statecharts (and, thus, conversation policies) represent a state's entry actions. In this case when the system enters the contract process state, it sets `require-ack` to `true`. For the rest of this explication I assume that these acknowledgment messages are being properly sent, received, and processed.

The supplier, having received the request for bid (Message #1), takes transition 10 in the bottom half of the statechart. Taking this transition fires Agent S1's "evaluate" event. The agent evaluates the request to send a bid. How this evaluation is to proceed is not specified by the conversation policy. It is not observable by Agent C; its inner workings do not affect Agent C;

and it does not need to be known by Agent C. The evaluation function would have to accept arguments with the following form:

(1) `evaluate(request, send(s1, c, msg(s1, c, offer, [...])))`

Thus, Agent S1 will evaluate a request that Agent S1 send an offer to Agent C on a certain item (described above in Message #1). Assume that the output of this evaluation is a term whose internal representation is

(2) `determination(msgID(c45), desired)`

This assumption does not affect the generality of this discussion; it simply provides specifics we can use to proceed.

Whenever an evaluation is completed, it sends the event `evalDone`. This triggers the statechart to attempt to take the two transitions leading out of "supplier knows" in the bottom half of the statechart. Since the value was determined to be "desired," the system takes the top transition. In doing this the system sends the events that cause the system to set the bid and then send it back to Agent C. The message shown in Figure 7 contains the

```
send(s1, c,
  msg(s1, c, offer,
      [reserve(z),
       Agent(z, s1),
       Object(z, x),
       room(x),
       beginDate(z, time(1999, 6, 6)),
       endDate(z, time(1999, 6, 10)),
       location(x, delft),
       price(x, $200)],
      [convID(v423),
       msgID(s57)]))
```

Fig. 7. Message #3

information from the request for an offer message plus the `price()` predicate.

The contractor receives this offer. It determines that this message belongs to conversation `v423` so it invokes the already-begun "contract process" conversation policy. It currently is in the "supplier knows" state. Having received the offer, it takes the transition to the "bid known" state and simultaneously fires the `evaluate` function. The evaluate function would have to accept arguments with the following form:

(3) `evaluate(offer, [...])`

Thus, Agent C will evaluate an offer for a reservation of a room at $200. Again, when this evaluation is done, it sends the event `evalDone`. This triggers the statechart to attempt to take the transitions leading out of the "bid known" state in the top half of the statechart. Assume the value was determined to be acceptable. This directs the system to take the transition to the "contract

made" state. In doing this, the system generates the event that causes the message accepting the bid to be sent (as shown in Figure 8).

```
send(c, s1,
  msg(c, s1, accept, msgID(s57),
      [convID(v423),
       msgID(c48)]))
```

Fig. 8. Message #4

The supplier receives this acceptance. It determines that this message belongs to conversation `v423` so it, again, invokes the "contract process" policy. It currently is in the "contractor has bid" state. Having received the acceptance, it takes the transition to the "contract made" state. Since both basic contract process substates have completed, transitions 9 and 18 are taken; this completes both substates of contract process which means that the superstate is completed as well.

The above example demonstrates how a conversation policy can provide fairly straight-forward message processing. The complexity of message handling is dramatically lessened by the pre-defined structure. One cost of this simplification is the added requirement that the conversation partners have to agree beforehand that they will use this policy. Further, for them to agree to use this policy it must already be defined and they must both have access to it. This is a difficulty that grows increasingly difficult with the number of conversation partners and with the types of conversations the agent engages in.

2.3 Inference-Based Message Handling

For a moment let us consider how the system can process messages without defining conversation policies. Assume that Agent B receives the message shown in Figure 9. This is a request from Agent A to Agent B that Agent

```
msg(a, b, request,
  send(b, a,
    msg(b, a, inform, [x]:name(b, x))),
  [ackPolicy([a, b], [parse], yes),
   timeSent(time(1999, 5, 12)),
   msgID(a63)])
```

Fig. 9. Message #5

B inform Agent A of Agent B's name (more directly, "`x` such that `x` is the

name of Agent B"). According to the definition of the standard effects of request [Moo01], the terms in Figure 10 show the results of this message. This specification indicates to the developer of Agent B that Agent A wants

```
considerForKB a wants do(b, send(b, a,
    msg(b, a, inform, [x]:name(b, x)))
considerForKB a wants (b wants do(b, send(b,
    a, msg(b, a, inform, [x]:name(b, x))))
```

Fig. 10. Results of a request to inform

the effects on Agent B of this message to be the following two items: 1) Agent B should consider adding to its knowledge base that Agent A wants Agent B to send a message to Agent A informing it of Agent B's name; 2) Agent B should consider adding to its knowledge base that Agent A wants Agent B to want to send that message. Because of the diversity of agent models, it is basically impossible to provide a general accounting of what the agent would do in response to this message and these two items in particular. What I provide below is an accounting of two reasonable approaches to defining a general means of handling these demands.

Simple agent What I am describing here is how a simple agent might implement the `receive` function when this "request to inform" message is received: Little about the agent model can be assumed other than the agent's

```
receive(msg(a, b, request,
  send(b, a,
    msg(b, a, inform, [x]:name(b, x))),
  [ackPolicy([a, b], [parse], yes),
   timeSent(time(1999, 5, 12)),
   msgID(a63)]))
```

Fig. 11. Message #6

ability 1) to receive FLBC messages, 2) to interpret them, and 3) to use the FL-SAS [Moo01] to initially process the message. In the FL-SAS process, the agent verifies that the message is valid and well-formed, composed of terms it knows the meaning of. It then ensures that it knows all the referents in the content of the message. In this message, the only referent is "b", the recipient of the message. Agent B next verifies that the message's content is compatible with its force; that is, it ensures that it is possible to "request" to "inform about the name of b." Finally, it ensures that the message as a whole makes sense.

```
receive(msg(From, b, request,
      send(b, From,
           msg(b, From, inform, WhatInfo),
      Context))) :-
  rightToKnow(From, WhatInfo),
  determineAnswer(From, WhatInfo, Answer),
  createNewContextTerm(Context,
       responseTo, NewContext),
  send(b, From,
       msg(b, From, inform, Answer),
       NewContext).
```

Fig. 12. Processing a request to inform with a simple agent

After the above steps are completed for this message (as they are for *all* incoming messages), Agent B is reasonably sure that it can process this message. This should simplify later processing of this message in something like the same way that both compiling a Java program simplifies the process of executing that program and checking XML messages to see if they are well-formed—these remove, early in the process, whole groups of errors. The agent can now get on with the process of completing that part of the process defined by the `considerForKB` items above. These are not meant to be directly executed because so little can be assumed about the agent model. Thus, these two `considerForKB` statements tell the developer of the receiving agent what the message sender meant to convey with the message but does not tell him or her how to implement the `considerForKB` function nor how to respond to the message. In this simple agent the developer has chosen 1) to not maintain a model of the sending agent's intentions, and 2) to do what the sending agent wants the receiving agent to do. Figure 12 shows a simplified Prolog representation of how this might be handled. The agent determines the recipient's right to know what it's asking, determines the answer to the question, creates a new context term (that might, for example, retain the conversation identifier and stack information, add a term indicating this is a response to a message, and add a new message identifier), and sends the reply.

More complex agent While the above addressed how a simple agent would respond to a request to inform, the following looks at how a more complex agent might handle this same message. The process for this agent begins as it did for the simple agent: the FL-SAS is applied with exactly the same steps. After this the actions of the agents diverge. In this agent the developer has chosen to implement a BDI (belief, desire, intention) agent model, to maintain a model of the sending agent's intentions, and to implement an agent that can make deductions about these intentions. A preferred agent could reason

```
receive(msg(From, b, request,
    send(b, From, msg(b, From, inform, WhatInfo),
       Context))) :-
  considerForKB(wants(From, do(b, send(b, From,
    msg(b, From, inform, [x]:name(b, x))))),
  considerForKB(wants(From, wants(b, do(b,
    send(b, From,
         msg(b, From, inform, [x]:name(b, x))))))).

considerForKB(wants(Other, do(b, Action))) :-
  /* do some processing, for example, the following */
  isBelievable(Other),
  worthRemembering(Action),
  addToKB(wants(Other, do(b, Action))).
considerForKB(wants(Other, wants(b, do(b, Action)))) :-
  /* do some processing, for example, the following */
  isBelievable(Other),
  worthRemembering(Action),
  addToKB(wants(Other, wants(b, do(b, Action)))),
  considerForIntentions(wants(b, do(b, Action))).

considerForIntentions(wants(b,
    do(b,
       send(b, From,
            msg(b, From, inform, WhatInfo),
            Context)))) :-
  /* do some processing */
  rightToKnow(From, WhatInfo),
  determineAnswer(From, WhatInfo, _),
  addToIntentions(do(b, Action)).

fulfillIntentions(do(b,
    send(b, From,
         msg(b, From, inform, WhatInfo),
         Context))) :-
  determineAnswer(From, WhatInfo, Answer),
  createNewContextTerm(Context, NewContext),
  send(b, From,
       msg(b, From, inform, Answer),
       NewContext).
```

Fig. 13. Processing by a complex agent

defeasibly about time and obligation (see [KM93a] for a discussion of what is needed for such a system and why it might be needed).

Figure 13 shows some snippets of Prolog code that might be used to reason about this message and how the agent would handle it. The `receive()` predicate is a fairly direct mapping from the terms shown in Figure 10. The first `considerForKB()` term specifies the restrictions on adding information about another agent's beliefs to the knowledge base. The second one does the same but also begins the process of determining (by invoking `considerForIntentions`) if the receiving agent's intentions should be affected by this incoming request. The agent's intentions are only affected if the requesting agent has a right to know the information and if the receiving agent can actually answer the question. (Of course, all these predicates could be more complicated and effective. For example, maybe the receiving agent cannot determine the answer but knows someone who might know the answer—in some cases this might be sufficient and appropriate.) Finally, the `fulfillIntentions` term would be called by some process when the agent is attempting to do what it desires to do.

Summary I have described agents with two different agent models. Each uses the FL-SAS to handle the initial message processing, and each interprets FLBC messages. Other than these similarities, the underlying structure of the agents differ. However, as the sketchy existence proofs I have given in these two sections indicate, each agent will create a similar response to the incoming message, and each was able to process the message without a predefined conversation policy.

2.4 Exception Handling

Another cost of the use of conversation policies (expressed as statecharts) versus inference-based methods is that—this is not going to be a surprise—they limit what can be said in a conversation. The standard conversation policy limits what messages can be said and when; however, there are some methods available for handling unexpected messages while still using CPs [Moo00b].

Consider again Message #1 (in Figure 5), in which the contractor requests that the supplier send an offer for a room reservation. This is the first message in the contract process CP. In response to this message the supplier begins to evaluate what it should do. The form of this predicate is shown in item (1) in §2.2. The full predicate might look something like that shown in Figure 14. This predicate has three main clauses. The first does some processing if it is able to determine that the agent desires to make an offer. If it determines that it is desireable, then the process would follow transition 11 in Figure 3. The second clause does something else if it is able to determine that it does not want to make an offer; this would send the process down transition 16 in

```
evaluate(request,
      send(s1, c, msg(s1, c, offer, Description))) :-
  ((/* some processing if desireable */)
   ;
   (/* some processing if unwanted */)
   ;
   (/* some processing if not enough information */
    not contains([beginDate, endDate, beds], Description),
    send(s1, [c], msg(S, R, request,
          send(R, S, msg(R, S, inform,
                [b]:[reserve(z), Agent(z, s1), Object(z, x),
                     room(x), beginDate(z, time(1999, 6, 6)),
                     endDate(z, time(1999, 6, 10)),
                     location(x, delft), beds(x, b)])),
          [cp(contractProcess), convID([v441, v423]),
           subordinate, timeSent(time(1999, 5, 11)),
           msgID(s62)])))).
```

Fig. 14. Evaluation that sends unexpected message

Figure 3. The last clause, and the one that is elaborated on, does some basic checking and determines that the description of the offer does not contain enough information to determine the attractiveness of the request. Suppose that the `evaluate` predicate ends up determining that more information than is provided in the incoming request is needed for the agent to reach a conclusion. In this case the predicate specifies that the agent should send a message back to Agent `C` requesting that it inform the supplier about the number of beds in the room the agent wants.

Agent `C` receives this message, examines it, and determines that it is part of the `contract process` it is already involved with. It also notes (from the incoming message's context term) that this message is part of a subordinate conversation and that this message does not match any of the events coming out of the "supplier knows" state in the top half of the contract process statechart. Using the technique for handling unplanned subdialogs with conversation policies discussed in [Moo00b], the system can use inferential processing (as described in §2.3 and §2.3) to link the CP described in §2.2 with another CP that is already defined for handling this request for information. The agents would go through the process of asking for and receiving the appropriate information. Having completed this subdialog, Agent `S1` would re-start the `evaluate` predicate shown in Figure 14. From the additional information it received in this last message, the agent should be able to come to a determination as to whether or not it wants to make an offer and, thus, continue with the conversation policy.

3 Conclusion

The above demonstration provides reason for believing that the system for managing conversation policies described in this paper can be integrated with an inference-based system for interpreting messages. Certainly, the contract process conversation policy could have been re-defined to handle the request for information discussed in §2.4; however, that is not really the point. If that message were the only one that might be sent in violation of the conversation policy, then of course the policy should be revised. Unfortunately, many such messages might occur and it is not at all certain that all these messages might be foreseen. Further, if these exceptions were built into the policy then the policy would become ever-more complex and difficult to implement. And all for the benefit of exceptions that may not ever be seen by the agent. The ability to gracefully handle exceptions allows conversation policy definitions to describe a flow of conversation directly without needlessly focusing on the myriad strange twists and turns it might take.

End notes Much of the work described in this paper was produced for the "Workshop on Formal Models for Electronic Commerce," organized by Yao-Hua Tan and Ronald M. Lee, and held at the Erasmus University Research Institute for Decision and Information Systems (EURIDIS), Rotterdam, The Netherlands, June 2–3, 1999.

References

[BDBW97] Jeffrey M. Bradshaw, Stewart Dutfield, Pete Benoit, and John D. Woolley, *KAoS: Toward an industrial-strength open agent architecture*, Software agents (Jeffrey M. Bradshaw, ed.), AAAI Press, 1997, pp. 375–418.

[BRJ99] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language user guide*, Addison-Wesley, 1999.

[DG00] Frank Dignum and Mark Greaves (eds.), *Issues in agent communication*, Lecture notes in computer science, vol. 1916, Springer, 2000.

[Fou97] Foundation for Intelligent Physical Agents, *FIPA 97 specification part 2: Agent communication language*, November 28, 1997, Geneva, Switzerland.

[GHB99] Mark Greaves, Heather Holback, and Jeffrey Bradshaw, *What is a conversation policy?*, Proceedings for the Workshop on Specifying and Implementing Conversation Policies (Seattle, WA) (Mark Greaves and Jeffrey Bradshaw, eds.), Autonomous Agents '99, May 1, 1999, pp. 1–9.

[Har87] David Harel, *Statecharts: A visual formalism for complex systems*, Science of Computer Programming **8** (1987), 231–274.

[KM93] Steven O. Kimbrough and Scott A. Moore, *On obligation, time, and defeasibility in systems for electronic commerce*, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III, Information Systems: DSS/Knowledge-Based Systems (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1993, pp. 493–502.

[LA87] Diane J. Litman and James F. Allen, *A plan recognition model for subdialogues in conversations*, Cognitive Science **11** (1987), 163–200.

[LF97] Yannis Labrou and Tim Finin, *A proposal for a new KQML specification*, Downloaded from `http://www.cs.umbc.edu/kqml/` in January 1998 (Technical Report CS-97-03), February 3, 1997.

[Moo00] Scott A. Moore, *On conversation policies and the need for exceptions*, Issues in Agent Communication (Frank Dignum and Mark Greaves, eds.), Lecture Notes in Artificial Intelligence, vol. 1916, Springer, 2000, pp. 144–159.

[Moo01] ———, *A foundation for flexible automated electronic commerce*, Information Systems Research **12** (2001), no. 1, 34–62.

[Pol88] Livia Polanyi, *A formal model of the structure of discourse*, Journal of Pragmatics **12** (1988), 601–638.

[RJB99] James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language reference manual*, Addison-Wesley, 1999.

Part IV

Agents and Strategic Interactions

Investigating the Value of Information and Computational Capabilities by Applying Genetic Programming to Supply Chain Management

Scott A. Moore[1] and Kurt Demaagd[2]

[1] University of Michigan Business School, Ann Arbor, MI, USA, samoore@umich.edu
[2] University of Michigan Business School, Ann Arbor, MI, USA, demaagdk@umich.edu

Abstract. In this paper we describe a research project centering on experiments in which game-playing evolving agents are used to investigate the value of information. Specifically, in these experiments we define populations of agents whose strategies evolve towards those that have better restocking strategies for their supply chain. The agents evolve their strategies in order to minimize costs (either for themselves or for their value chain). We describe several different experiments in which we will vary the abilities of agents both to gather and to store more information. Part of the results of this project will be related to the value of information and computational capabilities: Is it always better to have more information? If not, what are the conditions under which less information is better? The culminating experiment is one in which evolving agents compete to sell information to other evolving agents playing their roles in a supply chain.

1 Introduction

It is generally believed that having more information available to make a decision is a good thing; however, in today's computing-saturated business world information overload eventually occurs. Further, given that it costs money to manage information, care should be taken when the decision is made to gather an additional piece of information. The general strategy we are using in the research program described in this paper is to simulate a business problem, varying parameters so that players are placed under differing requirements for, or conditions of, information and computational capabilities. (For the rest of the paper, when it will not cause confusion, we will use the term "information" to refer to "information and computational capabilities.") The players who participate in these simulations will be created computationally through the use of a genetic program [Koz00]. Genetic programming is a computational approach to managing the evolution of agents whose fitness to live is based on their ability, relative to other agents in the

population, to perform in a specified way, for example to navigate a space or to compute a particular function. We are using genetic programming in this experiment as a means for investigating the information needs of the participants — the abilities both to gather and to make use of information — as they vary with the simulation's complexity. At the end of the series of experiments we will also look into how effectively evolving agents are able to sell information to the agents competing in the supply chain. A secondary result of this research program will be an increased understanding of how sensitive the genetic program is to its initial parameters, evidence showing how varying certain genetic program-related parameters affect the experimental results, and how well a genetic program can navigate a large search space. The specific business situation we will use in this research program is a multi-level supply chain game as exemplified by Sterman's beer game [Ste89]. Throughout the rest of the paper we use the term *beer game* to refer to the simulation we are investigating. A final result of this project will be insight into how well solutions found by a genetic program compare with solutions found by other methods in the management science and information systems literature. This paper describes the research program we are undertaking on this subject.

In the initial experiments described in this paper, agents evolve in several separate populations that correspond to the different roles in the beer game, namely retailer, wholesaler, distributor, and manufacturer. (See Figure 1.) The standard scenario is one in which an agent can only send orders to

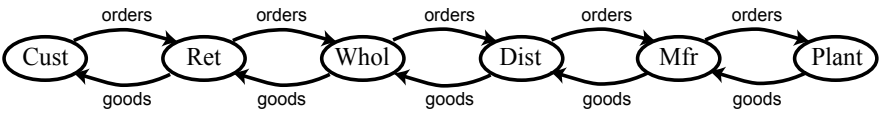


Fig. 1. Basic set-up of supply chain

one agent who is one step upstream (referring to the flow of goods) and can only send goods to one agent who is one step downstream. Demand is determined by an exogenous customer who both sends orders to the retailer and ultimately receives final delivery of the goods. The times for orders and goods to arrive at their destinations are taken from distributions that are not known by the players. The goods are manufactured at the plant, another exogenous player.

Each week each player has a standard series of tasks that it completes (see Figure 2). First, goods arrive from the player upstream; these goods are put into inventory. Second, orders arrive from the player downstream; these orders are added to any previous backorders (i.e., orders from past weeks that have not been filled). Third, the player ships as many goods as it can in order to fulfill the orders it has received. Fourth, the player places an order

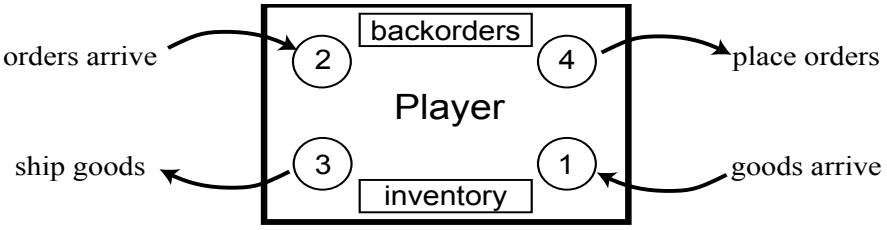


Fig. 2. A player's tasks each week of the game

for the number of goods that it thinks it needs. It is in this last step that the player's re-stocking strategy is applied.

The object of evolution on these populations is to find a strategy for ordering goods, for each player, that minimizes its costs when playing the game with other agents. People are notoriously bad at playing this game even at the simplest of possible settings (e.g., when demand, lead time, and so on are more or less fixed at constant values). An average player's performance has been known to end up with costs ten times worse than optimal [Ste04]. Further, analytical techniques can provide solutions to this problem only when the settings are simple, but not when demand, lead time, etc. stochastic and non-stationary, as they usually are in real environments [Che99].

We have several research questions in mind when looking at what evolves in this environment, some related to the genetic programming tool itself and others related to the supply chain management process. We will not summarize them all, but there are generally two classes of questions. The first relates to determining if a genetic program will be able to evolve agents that use optimal or near-optimal re-stocking strategies. We will be playing a great variety of supply chain games, varying many parameters, including the number of players at each level of the supply chain, the information available to the players, the speed at which goods are shipped, and the pattern of customer demand. We believe that some scenarios will prove to be quite simple for the genetic program while others will prove more demanding. Kimbrough & Wu [KWZ02] successfully used genetic algorithms (a related technique) to evolve agents to play this game, though the search space for their solutions is much constrained relative to our proposal.

The second set of questions relate to looking at the value of information. We will be looking for situations in which one piece of information ends up determining a player's eventual success. Conversely, we will also be looking for situations in which the volume of information overwhelms a population and keeps any one member from finding successful strategies. We will also be looking for ways we can value information. In one set of experiments we will put costs on the usage of certain information in order to see how the population reacts. In another set we will add a competing population that evolves pricing strategies for information in order to determine if the pric-

ing agents can determine appropriate and sustainable prices for information without adversely affecting the players' performance.

We wrote the program that runs the simulations for this project using GNU CLISP [Fre04]. The source code for our program is available at SourceForge [MD04] and runs on Windows, Linux, and Unix machines. In its current state of development it is able to generate the populations, play the games with a limited set of parameters, breed a new generation based on the performance statistics of the existing generation, and generate some statistics while it is running. Much remains but the foundation of the program is completed. More information about the computing resources used in this simulation are contained in Appendix C.

2 The Evolutionary Process

In this section we describe the basic set-up of the evolutionary process, the game played by the members of the population for the purposes of determining the fitness of each population member, and the representation of each player's re-stocking strategy.

2.1 Basic Set-Up

We will vary the experiment somewhat throughout this project but the following describes a standard set-up. (See Figure 3.) This experiment has Φ separate, co-evolving populations of Λ_ϕ agents for each role in the game; in Figure 3 the dots in the circles represent the members of the population.

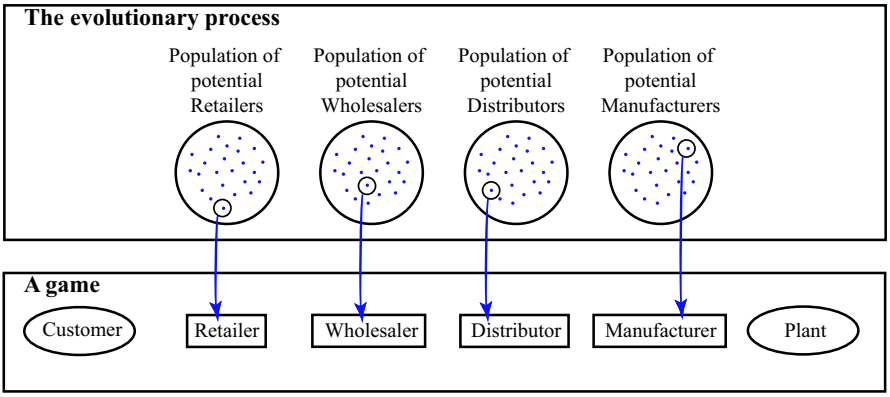


Fig. 3. The relationship between the evolutionary process and the supply chain games

(The overall evolutionary scenario is summarized in Figure 4, and related term definitions are shown in Figure 5.) For each evolutionary scenario the

```
Create each population of agents
gen = 1
repeat
   Instantiate a particular demand pattern
   for φ=1 to Φ
      Choose agent j from φ at least M times
         Fill all the roles in the game appropriately
         Play the game for W weeks
         Record the outcome for each agent
      end-loop
   end-for
   Calculate and record fitness scores for all agents
   Determine this generation's best agent
   Determine the scenario's current champion so far
   Breed the next generation of each population
   gen = gen + 1
until termination condition is met
```

Fig. 4. Evolutionary scenario

researcher defines a particular demand distribution (e.g., uniform over a certain range, Poisson with a particular arrival rate). For each generation, a particular instantiation of this particular demand distribution is created. In each generation of the process, each agent plays at least M games (this is a parameter set by the researcher), and it is the agent's performance in these games that determines its fitness. For any one of these games, a random member of each population is chosen to play its specified role (as shown in Figure 3). At the end of the game, the agent's score is recorded. We use several different fitness functions but a standard one is to minimize an agent's costs. One game is usually played for $W = 35$ weeks (i.e., *turns*), though we will vary this to keep the agents from over-specifying. After all games are played, the program calculates and records fitness scores for all the agents in all the populations. From these scores the generation's best agent is determined. Determining the "best" agent is something of a difficult task in this experiment and is covered in some depth in §2.2. After determining the generation's best agent, the program determines if this agent is better than the best agent that the scenario has uncovered so far. Information about the current champion (as we refer to this overall best agent) is retained and updated after each generation. The best players in each generation are then chosen to reproduce and create the next generation of players (using various approaches). This entire process is repeated until a termination condition is met, typically a completion of a specified amount of simulated time.

2.2 Measurements

In the next several sections we are going to discuss the many measurements that are made in this program. We start by describing many of the basic variables and functions that are needed to make the measurements discussed in the following sections. This supply chain game lends itself to many different ways of keeping score; we describe several of them in §2.2. The genetic programming process requires that a specific score be kept as a means of guiding the process of reproduction. This score is called the *fitness function*; we describe this in §2.2. We conclude (§2.2) with a discussion of how the genetic program itself can be measured.

Useful variables and functions The basic parameters shown in Figure 5 are needed for the following discussion. (Note that all the terms defined in

$$
\begin{aligned}
\Phi &= \text{number of populations} \\
\phi &= \text{any value in } 1 \ldots \Phi\text{; that is, a particular population} \\
\mathcal{P}_\phi &= \text{a certain population } \phi \\
\Lambda_\phi &= \text{number of agents in population } \phi \\
j &= \text{a particular member of some population} \\
\Lambda &= \text{number of agents in all populations} \\
&= \sum_{\phi \in \Phi} \Lambda_\phi \\
\mathcal{R} &= \text{the roles in a game: retailer, wholesaler, distributor,} \\
&\quad \text{and manufacturer} \\
\rho &= \text{any value in } \mathcal{R}\text{; that is, a particular role} \\
W &= \text{number of weeks} \\
w &= \text{any value in } 1 \ldots W\text{; that is, a specific week} \\
G &= \text{number of games played in a particular generation} \\
g &= \text{any value in } G\text{; that is, a particular game} \\
M &= \text{the minimum number of games an agent plays in one generation} \\
H_{g,\rho} &= \text{inventory holding costs per item in game } g \text{ for the role } \rho \\
P_{g,\rho} &= \text{penalty cost per backordered item in game } g \text{ for the role } \rho
\end{aligned}
$$

Fig. 5. Definitions of standard parameters

the following figures refer to one generation; for example, G refers to all the games that occur in one generation.) These are set before an evolutionary scenario is begun. Consider the first function, $\pi(g, \rho)$. This returns the index

$\pi(g, \rho)$ = the index j of the player who played role ρ in game g
$\phi(j, \rho)$ = the set of game numbers in which player j played role ρ
$\mu(j, \rho)$ = the number of games in which player j played role ρ
$\psi(\rho)$ = the population number that players in role ρ are taken from

Fig. 6. Useful functions

j of the player who played role ρ in game g. This function enables us to tie the score earned by a player in a role back to a specific member j of the population — which, in turn, enables us to assign fitness scores and then reproduce a new generation of players. You will see this function π used in many variables. The function $\phi(j, \rho)$ is basically the inverse of the π function: given the index j identifying the population member and the specific role, the function returns a set of game numbers for which this agent fulfilled this role. The next function, $\mu(j, \rho)$, returns the number of games in which player j played role ρ. This will end up being different values for different members of the population in a particular generation since members are randomly selected for games.

The last function in this table, $\psi(\rho)$, is useful when there is more than one population of agents (as there is in the scenario we are currently examining; see Figure 3; if there were only one population, then this function would always return 1). In our conception of this genetic programming investigation, at the beginning of the evolutionary scenario each population of agents is created and assigned to a particular role in the game. Thus, if you know the role in the game, you know the population that any player in that role came from. This function captures this information.

Figure 7 describes important weekly status variables for the supply chain game and how they are related to each other. These values are calculated and stored for each player for each week for each game for each generation. In most games the information for a particular player will be available to only that player. In some games the player will know information about all prior weeks of the game while in other games the player will only know about the current week. All but the last two variables in this figure are relatively straight-forward. The last two are related to the problem of coming up with a measure for the whole value chain. The first, $K_{g,w}$, defines a simplistic way of calculating the costs for a value chain for one week of a game: add up everyone's cost. Consider, though, if all of the members of a value chain were under one management so that both inter-agent backorders and the location of inventory were irrelevant; if this were the case, then $\kappa_{g,w}$ would be the preferred measure.

$$
\begin{aligned}
B_{\pi(g,\rho),w} &= \text{inventory on hand at beginning of week } w \text{ for the player} \\
&\quad \text{in role } \rho \text{ in game } g \\
S_{\pi(g,\rho),w} &= \text{shipment received in week } w \text{ by the player in role } \rho \text{ in game } g \\
D_{\pi(g,\rho),w} &= \text{demand received (that is, the number of goods ordered)} \\
&\quad \text{in week } w \text{ by the player in role } \rho \text{ in game } g \\
I_{\pi(g,\rho),w} &= \text{inventory position in week } w \text{ for the player in role } \rho \text{ in game } g \\
&= B_{\pi(g,\rho),w} + S_{\pi(g,\rho),w} - D_{\pi(g,\rho),w} \\
O_{\pi(g,\rho),w} &= \text{backordered items in week } w \text{ for the player in role } \rho \text{ in game } g \\
&= \begin{cases} 0 & \text{if } I_{\pi(g,\rho),w} \geq 0 \\ -I_{\pi(g,\rho),w} & \text{if } I_{\pi(g,\rho),w} < 0 \end{cases} \\
C_{\pi(g,\rho),w} &= \text{costs in week } w \text{ for the player in role } \rho \text{ in game } g \\
&= \begin{cases} I_{\pi(g,\rho),w} \times H_{g,\rho} & \text{if } I_{\pi(g,\rho),w} \geq 0 \\ O_{\pi(g,\rho),w} \times P_{g,\rho} & \text{if } I_{\pi(g,\rho),w} < 0 \end{cases} \\
K_{g,w} &= \text{total costs in game } g \text{ in week } w \text{ for a value chain} \\
&= \sum_{\rho \in \mathcal{R}} C_{\pi(g,\rho),w} \\
\kappa_{g,w} &= \text{aggregate costs in game } g \text{ in week } w \text{ for a value chain} \\
&= P_{g,\rho} \times O_{\pi(g,R),w} + \sum_{\rho \in \mathcal{R}} (H_{g,\rho} \times I_{\pi(g,\rho),w})
\end{aligned}
$$

Fig. 7. Important status parameters for a week in a game

Keeping score in a game We now have some idea about status information that is available during a game. In this section we define several different ways to keep score in a game. Figure 8 defines for agents various game scores whose values are based on these variables. $A_{\pi(g,\rho)}$ defines the total costs for the player in role ρ in game g; it is the sum of the player's costs (defined in Figure 7) for each week of the game. Take a closer look at the name of this variable: $A_{\pi(g,\rho)}$. This uses the function π that we discussed above. When we have a value for a specific $A'_{\pi(g',\rho')}$ — that is, when we are considering a specific role ρ' in a specific game g' — the π function tells us the specific member of the population j' that played in the game; thus, $A'_{j'}$ represents the total costs that player j' incurred in that role in that game.

Depending on the demand function used when running the evolutionary process, it is possible for players in the same population to play games with different demands in different generations. As a result, it can be difficult to use A to compare the fitness scores between different generations. For example, players in two different generations might play similar games except that the first player faces an average demand of 10 while the second player faces an average demand of 100. Both players may have sub-optimal strategies — for

$$
\begin{aligned}
A_{\pi(g,\rho)} &= \text{total costs for the player in role } \rho \text{ in game } g \\
&= \sum_{w=1}^{W} C_{\pi(g,\rho),w} \\
\mathcal{D}_{g,\rho} &= \text{total demand in game } g \text{ for role } \rho \\
&= \sum_{w=1}^{W} D_{\pi(g,\rho),w} \\
\Upsilon_{\pi(g,\rho)} &= \text{normalized total costs for the player in role } \rho \text{ in game } g \\
&= \frac{A_{\pi(g,\rho)}}{\mathcal{D}_{g,\rho}} \\
A^{\Xi}_{\pi(g,\rho)} &= \text{the total costs for a player in role } \rho \text{ facing the demand} \\
&\quad \text{pattern used in game } g \text{ if its strategy is to always order} \\
&\quad \text{according to the presumably standard strategy } \Xi \\
\eta_{\pi(g,\rho)} &= \text{relative total costs for player } \rho \text{ in game } g \\
&= \frac{A_{\pi(g,\rho)}}{A^{\Xi}_{\pi(g,\rho)}}
\end{aligned}
$$

Fig. 8. Various game scores for an agent in a game

example only ordering 50% of demand — but the second player will appear to have a worse strategy. It will probably have backorder costs that are a factor of 10 higher than the first player's. Therefore, we need a way of comparing one agent's score with the score of another agent who faced a different set of demands.

The *total demand normalized total costs for a player*, $\Upsilon_{\pi(g,\rho)}$, addresses this issue. Υ is defined as a player's total costs divided by the sum of its weekly demands. The resulting game score represents a player's cost per unit of demand. Returning to the previous example, the first player had a backorder cost of 5 and a demand of 10, resulting in a Υ of .5. Likewise, player two had a backorder cost of 50 and a demand of 100, resulting in a Υ of .5. This Υ, which we shall refer to as TD-normalized, partially resolves the problem of differences in demand artificially skewing the fitness scores.

What $\Upsilon_{\pi(g,\rho)}$ does not address is the difficulty raised related to the fact that a player's actions (and resulting score) can be non-linearly related to the incoming demand. What is needed is a score for an agent that uses some sort of "standard" strategy; this score could be used as a baseline against which to measure the quality of a strategy. In this case $A^{\Xi}_{\pi(g,\rho)}$ is our baseline, where we will assume for now that $\Xi = D$. The D strategy (placing an order equal to the player's demand) is a simple and intuitively appealing baseline strategy which is also the optimal strategy in simplistic scenarios (as pointed out in [KWZ02] among others). In some cases we will be able to use as the baseline

the optimal strategy as defined by the operations management literature but in other cases we will use heuristics (such as the D strategy).

Given this baseline, it is possible to measure the *relative average fitness*, η; it is $A_{\pi(g,\rho)}$ divided by the baseline. The resulting ratio will give a comparison of the strategy's effectiveness versus the baseline D strategy. Of course, in different scenarios it might be useful to use baseline strategies other than D. The measure η will provide a convenient way to compare strategies across generations while also signifying how well the genetic program is progressing relative to some known standard.

In addition to looking at the performance of individual players, we also can examine the performance of the entire supply chain for a whole game (see Figure 9). There are two ways to do this, one based on T_g and the

$$
\begin{aligned}
T_g &= \text{total costs in game } g \text{ for a value chain} \\
&= \sum_{w=1}^{W} K_{g,w} \\
U_g &= \text{TD-normalized total costs for the value chain in game } g \\
&= \frac{T_g}{\mathcal{D}_{g,R}} \\
T_g^{\Xi} &= \text{the total costs for a value chain in game } g \text{ if the strategy of} \\
&\quad \text{each player is to order according to the standard strategy } \Xi \\
N_g &= \text{relative total costs for the value chain in game } g \\
&= \frac{T_g}{T_G^{\Xi}} \\
\Theta_g &= \text{total aggregate costs in game } g \text{ for a value chain} \\
&= \sum_{w=1}^{W} \kappa_{g,w} \\
\mathcal{U}_g &= \text{TD-normalized aggregate total costs for the value chain in game } g \\
&= \frac{\Theta_g}{\mathcal{D}_{g,R}} \\
\Theta_g^{\Xi} &= \text{the aggregate total costs for a value chain in game } g \text{ if each} \\
&\quad \text{player orders according to the standard strategy } \Xi \\
\mathcal{N}_g &= \text{relative aggregate total costs for the value chain in game } g \\
&= \frac{\Theta_g}{\Theta_g^{\Xi}}
\end{aligned}
$$

Fig. 9. Various scoring choices for a value chain in a game

other on Θ_g. These, in turn, are based on the two different ways in which we measure weekly value chain performance, $K_{g,w}$ and $\kappa_{g,w}$, respectively.

Each of these measurements of total value chain performance (i.e., T and Θ) has the following associated values: 1) TD-normalized scores (U_g and $\mathcal{U}_g$, respectively), 2) baseline scores (T_g^{Ξ} and Θ_g^{Ξ}, respectively), and 3) relative scores (N_g and $\mathcal{N}_g$, respectively). The motivation for these TD-normalized and relative scores mirrors that of the TD-normalized (Υ) and relative (η) scores described above.

Keeping score for reproduction We can monitor the genetic programming evolutionary process with many different scores, but for purposes of reproduction we have to select one and only one score. Figure 10 describes several fitness measures as defined by Koza [Koz00]. The simplest measure

$$
\begin{aligned}
\tau_{\pi(g,\rho)} &= \text{game-based raw fitness; this can be any one of } A, T, \text{ or } \Theta \\
\overline{\tau}_j &= \text{an agent } j\text{'s average raw fitness over all of its} \\
&\quad \text{games in this generation} \\
&= \frac{1}{\mu(j,\rho)} \sum_{g \in \phi(j,\rho)} \tau_{\pi(g,\rho)} \\
\beta_\rho &= \text{best score for any player in role } \rho \text{ in this generation} \\
&= \min_{j \in \mathcal{P}_{\psi(\rho)}} \overline{\tau}_j \\
\sigma_{j,\rho} &= \text{standardized fitness for the player } j \text{ in role } \rho \text{ in game} \\
&\quad g \text{ where } \pi(g,\rho) = j \\
&= \overline{\tau}_j - \beta_\rho \\
\alpha_{j,\rho} &= \text{adjusted fitness for agent } j \text{ in role } \rho \\
&= \frac{1}{1+\sigma_{j,\rho}} \\
\nu_{j,\rho} &= \text{normalized fitness for agent } j \text{ in role } \rho \\
&= \frac{\alpha_{j,\rho}}{\sum_{j \in \mathcal{P}_{\psi(\rho)}} \alpha_{j,\rho}}
\end{aligned}
$$

Fig. 10. Koza's standard fitness measures

is τ, the game-based raw fitness. The value of this can come from the fitness measures defined in Figures 8 and 9, such as A, T, or Θ. (We do not use any of the TD-normalized or relative scores as a basis for a fitness measure, though we could, because they are transformations that reduce the absolute differences among scores; however, we will track the TD-normalized and relative scores as part of the reporting process for the reasons described above.) An agent's raw fitness for a generation, $\overline{\tau}_j$, can be a bit more complicated than simply using an agent's raw fitness score from one game. In many instances the agent has the opportunity to play multiple games. For example,

in the evolutionary scenarios in which there are multiple populations, we will sometimes have the agent play games with multiple different, random opponents. The hypothesis is that this will reduce the chances that a high-fitness agent will be dropped from the population because it played in a game with a poor value chain partner. For this reason we calculate an agent's fitness in a generation, $\overline{\tau}_j$, as that agent's average game score over all the games that it plays in that generation.

The genetic program needs standardized fitness, $\sigma_{j,\rho}$, to have values such that smaller numbers are better, and the best score is zero [Koz00, p. 96]. All of the measures defined in Figure 8 are based on costs, which should be minimized. Therefore, as lower is already better, standardized fitness is equal to raw fitness (with the β_ρ adjustor needed to set the best score to 0).

The adjusted fitness for the player, α, converts the standardized fitness into a number between 0 and 1. As in the case of the standardized fitness, this will result in a score where smaller values are better. It also tends to create a more pronounced distinction (relative to the distinction among standardized fitness scores) between scores with similar values. Normalized fitness, ν, is similar to the adjusted fitness, except larger scores are better. Like adjusted fitness, the values will range from 0 to 1 and it tends to further emphasize the difference between values.

When evolving agents, we will use all of the possibilities for the game-based raw fitness, τ, to calculate an agent j's score for any particular game, and will use the normalized fitness based on that τ, ν, as the agent's fitness value for that generation. We will report all of the τ values for every generation no matter whether or not that specific τ is being used as the basis.

Comparing scores across experiments The above sections describe a variety of measures which provide different types of insight into the performance of the player, the value chain, and the genetic program's ability to improve the strategy of the players in the game. In order to compare scores across experiments, the score has to be insensitive to the particulars of the demand pattern faced, the population size, the number of generations, and any of the game's settings. The purpose of this score is to compare the effectiveness of the genetic program (and the particular settings used in that particular experiment) at producing players who play the game well. The goals of these experiments are 1) to find the best re-stocking strategies for particular demand distributions, and 2) to find settings for the genetic program that most effectively and efficiently search for re-stocking strategies.

Before continuing, let us clarify that there are three *classes* of scores that we will be keeping: those based on a player's scores, those based on a value chain's total costs, and those based on a value chain's aggregate costs. These measure different aspects of a player's performance and are not just simple linear transformations of the others.

The basis of all the scores (within one *class*) is a player's total costs for a game, A. We defined the player's TD-normalized total costs, Υ, so that we can consider the player's score in a way that minimizes the effects of the scale of the demand placed on the player. We defined the player's relative total costs, η, so as to minimize both the pattern of the demand placed on the value chain and the rest of the game's parameters. However, it is still the case that a strategy might perform well against one demand instantiation but not against another. Recall that at the beginning of each generation we generate a new demand instantiation that is used for all games played in that generation. Our goal is to define a genetic programming process that discovers restocking strategies that are appropriate for certain demand distributions and not just for certain instantiations of those demand distributions.

It is also the case that we're interested in *all* of the strategies that the genetic program looks at in the course of the evolutionary process, not just the ones considered in the last generation. We want to know what the *best* strategy was over the course of the entire process, and we want the performance of the experimental settings to be derived from the the score of this player. The pertinent questions become "how do we determine what *best* means in this context?" and "what is the appropriate process for finding the best player?"

For our purposes, we are attempting to define a genetic programming process that finds a strategy that is most effective against *any* demand instantiation from a particular demand distribution. Each generation each member of a population plays games against the same demand instantiation; the next generation will use a different demand instantiation. This means that the "best" player of any one generation has the lowest total costs against the particular demand instantiation used in that generation. Over the course of an n generation experiment, members of the population face n different demand instantiations.

We propose the procedure described in Figure 11 for finding the best restocking strategy in an experiment. Each generation one player emerges as the best performer against the specific demand instantiation that was used during that particular generation. This player is that generation's best candidate to defeat the current champion. The *best* player from an experiment will be the current champion after the last generation.

This process best captures what we mean when we say that we are trying to find a strategy that performs the best against a certain type of demand distribution. In the long run we do not care whether or not a certain strategy performs the best against a particular demand instantiation — we are interested in those that perform the best against any demand instantiation from a particular demand distribution.

It should be noted that this process of choosing a champion depends on, but is not a part of, the genetic programming process. The experiment's best player does not even have to be in the population at the end of the experiment; it might have existed in an earlier generation but, through the

1. Generate a set of 30 different demand instantiations from the demand distribution used in the current experiment. Call this the *championship demand set*. If a player plays one game against each of the instantiations in the championship demand set, then it is said to have played the championship games.
2. Have the standard strategy play the championship games. Call the average of these scores the *standard championship score*.
3. After the first generation, save the player with the lowest total costs as the current *champion*.
4. Have the current champion play the championship games. Average these scores. Call this the *current champion's score*. Divide the current champion's score by the standard championship score; call this the *current champion's ratio*.
5. Play the next generation.
6. After this generation, the player with the best score for that generation will be named as the *challenger* to the current champion.
7. The challenger will play the championship games. The challenger's ratio for these games will be compared against the current champion's ratio. The player with the lowest ratio will be declared the champion.
8. Keep a record of the current champion's ratio.
9. Repeat starting at #5.

Fig. 11. Process for determining the best player over the course of an experiment

vagaries of the genetic programming process, might not have been selected to reproduce into the next generation. Further, although this process depends on the genetic programming process, it does not at all influence the course of the on-going evolution. The championship process merely captures valuable information that might have otherwise been lost during evolution.

2.3 Specifying a Strategy

We have discussed how the game is played, how the evolution proceeds, how agents are measured, and how experimental results are evaluated. We now examine how strategies are defined and how new ones evolve from old ones.

Defining a strategy Each member's strategy is represented by an S-expression. An S-expression is a set of terms composed into a tree. Consider the example shown in Figure 12. The standard mathematical expression can be translated in a fairly straight-forward manner into the S-expression; this can, in turn, be mapped onto a tree representation. The program that we have written manipulates S-expressions but the tree representation can sometimes be easier to use in exposition so we will generally use that representation in this paper. The terms in the S-expression can be either terminals or functions. Terminals are either constants or variables, such as 187 or D. A function takes a specified number of parameters, performs an operation on those parameters, and then returns the result. For example, the function `(sum 3 4)` will add 3

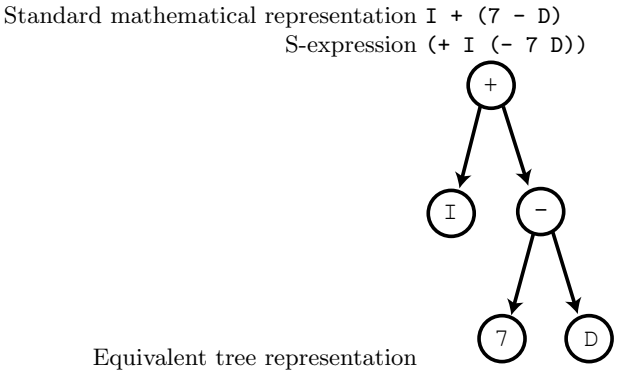


Fig. 12. Equivalent representations

and 4 and return the result. Parameters of a function can be both terminals and other functions.

When coming up with a set of functions to include in a genetic program, it is desirable that every function satisfy the *closure* property [Koz00, p. 86]. To meet this requirement, all functions must accept as a parameter any possible value in the set composed of all terminals and any value returned by any function. Thus, no matter what allowed transformations that might be performed on this function (see the discussion of genetic operations below), it should not encounter errors in execution. The classic example is division by zero. Since this would generate an error, a special division operator must be created to return a pre-specified value if the denominator is zero. In our experiments, if a function encounters an error condition, it will return 100,000, a number far out of the range of possibly useful order values.

Functions must also satisfy the *sufficiency* property if we want the genetic program ultimately to find the answer to the problem at hand. This states that functions and terminals provided must be able to express a solution to the problem. For example, in our experiments with the beer game and MIT demand, we know that the optimal solution is based on the demand. If we failed to provide the demand as a terminal, the program would at best be able to find only an approximation of the optimal solution.

The terminals and functions are all listed in §A. The terminals consist of a set of integers plus information that the agent knows about itself and the game: amount of its own inventory, amount that it has on order, its own current demand, plus the current week number of the game. There is nothing that says that the appropriate terminals or functions have to be easy to find, or that they have to be combined in some straight-forward way, or that the researcher even knows *how* the terminals and functions should be combined in order to reach the best solution. In fact, part of what we are attempting to do with this research program is to determine how difficult the answer can

be to find while still enabling the genetic program to succeed. Further, some of the problems that we will pose to the genetic program have no known optimal solution so we will merely be searching for "best that we can find." In this case, we will investigate the value of information and computational capabilities and how well the agents are able to take advantage of them.

The functions currently under consideration come in three groups (see Appendix A): mathematical, logical, and informational. The mathematical set includes fairly standard operations: addition, subtraction, multiplication, integer division, minimum, maximum, and sine. The logical functions are also the ones you might expect: and, or, not, if/then, if/then/else, greater than, less than, and equal to.

The informational functions are a bit more interesting. The function `(mydem y)` retrieves a player's own demand `y` weeks ago while `(myinv y)` does the same thing for inventory and `(myord y)` for orders. The functions `(dem x y)`, `(inv x y)`, and `(ord x y)` are generalizations of these three functions that provide access to this information for other players. These functions will not be available to all players in all scenarios. Some of our experiments will be directed at discovering how useful the players find these functions, and whether or not some functions allow some population members to dominate over other population members.

Evolving a new strategy Genetic programs generate new strategies using many of the same principles of Darwinian evolution. It is essentially a matter of selecting certain individuals with superior fitness, performing a set of genetic operations, and adding them to a new population of individuals. In this way, the genetic program should eventually evolve better solutions to the problem.

The first problem is to select superior individuals. In §2.2 we described the different means of rating an agent. The next problem is to select individuals with higher rankings without destroying the diversity of the genetic material that may contain components of the optimal solution. Three primary methods are used in genetic programming: fitness proportionate, rank, and tournament.

Fitness proportionate selection is the most popular method for selection [Hol92]. This method effectively treats the fitness score of the individual as the probability that it will be selected. The easiest means of calculating this probability for an agent j is to set it equal to the normalized fitness, $\nu_{j,\rho}$. The normalized fitness is already a measure between 0 and 1 with greater scores being better, and the sum of all νs is 1.0.

While popular, this has a significant weakness. If a particular agent or small group of agents has a relatively high fitness, they may quickly dominate the entire population. This can reduce the diversity of the population which may limit the ability to find an optimal solution. One means of resolving this is to use *rank* selection. Each individual is assigned a rank based on its

score, ν. New members of the population are selected from the set of highly ranked members. This has the effect of reducing the premature convergence of fitness proportionate selection while still favoring strong individuals.

Tournament selection is the computational equivalent of two individuals competing for a mate. Two agents are selected at random from the population. The agent with a superior fitness score is selected. Once the agent is selected, one of several genetic operations is performed on it. The goal of this phase is to create a new population that, ideally, will be superior to the previous population. Three of the most common operations used in genetic programming are reproduction, crossover, and mutation.

The simplest one of these is *reproduction*. The selected individual is simply copied into the new population. This results in the stronger elements of the previous population becoming the new population. While this will create a stronger population, the system will never find any members with superior fitness compared to those in the original population. To resolve this issue, members of the population must be modified in some way.

The most common method of evolving the individuals is *crossover*.

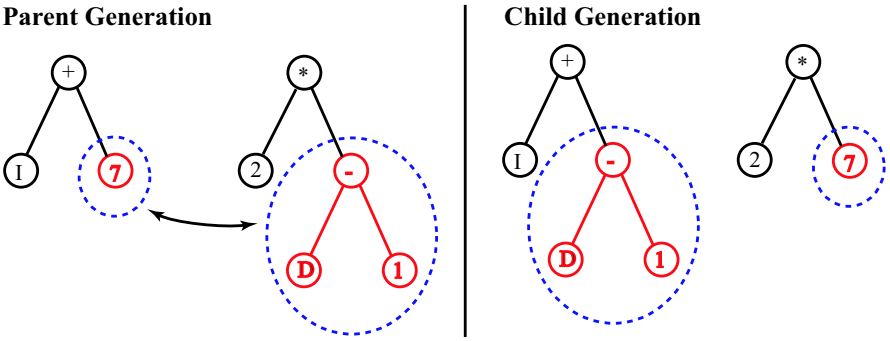


Fig. 13. An example of crossover

This is the genetic programming equivalent of sexual reproduction. It takes material from two parents and uses it to create two new children. The first step is to select two individuals, using one of the selection methods described above. These two agents will be the parents.

The next step is to perform the actual crossover. Recall that each parent can be represented as a tree (Figure 12). To perform the crossover, each parent swaps part of its tree with the other parent. This is done by selecting a random point in each parent. These points become the crossover points. The subtrees under these points are swapped between the two parents (see Figure 13). The resulting new trees are then inserted into the new population. It is possible to select the same tree twice so that it serves as both parents. In such a case, however, the resulting children will most likely not be clones

of the parents. To get a clone, the crossover operation would have to select the same crossover point on both trees. Since all points are equally likely to be selected, the probability of this occurring is minimal.

While the trees shown in Figure 13 are relatively simple, this same process occurs even with more complex trees. This crossover operation can grow and shrink the size of the tree; Figure 14 shows an arbitrary tree that was generated during the course of an experiment. In the event that the tree grows to a depth greater than the maximum depth allowed, the crossover operation is cancelled and the first parent is inserted into the new population. In this way crossover may mimic reproduction.

The final option is *mutation*. When a particular population appears to be prematurely converging on a solution, it may be desirable to inject some diversity into the system. As the title implies, mutation introduces a random change into the system. It does this by selecting a random point in a selected tree, removing that point and all elements below it, and then inserting a new random tree into its place. The crossover operation in genetic programming, however, is highly effective at finding new solutions. In addition, if a crossover point is selected on a terminal, it has the same result as a mutation. Therefore, mutation is very rarely used.

3 Plan of Investigation

In this research project we are completing four series of experiments that will generally build upon the results of each prior series. The first set of experiments will focus on fine-tuning the parameters of the beer game and the genetic program so that the evolution process will go better. The second set of experiments will focus on parameters of the genetic program that might significantly affect how well the evolutionary process progresses in our later, more complex, experiments. The third set of experiments are the centerpiece of our investigation into information value and information overload. The fourth set of experiments will build on this last set of experiments. We will use what we learn from the third set to define future investigations, but we list here a reasonable set of possibilities.

3.1 Given Parameters

In the following experiments we will use default values for some parameters: probability of permutation (set to 0) [Koz00, p. 107], probability of editing (0) [Koz00, p. 108], and the probability of encapsulation (0) [Koz00, p. 110]. Koza discussed each of these but used them little, if at all, in his experiments. The *variety* of a population is "the percentage of individuals for which no exact duplicate exists elsewhere in the population" [Koz00, p. 93]. Our program checks for duplicates so the variety will be 100%. We do this so as to ensure the greatest amount of genetic building blocks enter our population with the

```
(IFTHEN
  (SUM
    (SUM
      (DIFF
        (DIFF D
          (DIFF
            (SUM
              (IFTHEN 2
                (IFTHEN D
                  (IFTHEN (IFTHEN D D) 2)))
              3)
            (DIFF
              (IFTHEN
                (SUM
                  (SUM D D)
                  (DIFF
                    (IFTHEN
                      (DIFF D 2)
                      (DIFF (IFTHEN D D) 2))
                    2))
                D) 2)))
        (IFTHEN
          (SUM
            (IFTHEN
              (DIFF (IFTHEN D D) 2)
              (DIFF (IFTHEN D D) 2))
            2)
          (DIFF (IFTHEN D D) 2)))
      D)
    (DIFF
      (IFTHEN
        (DIFF D 2)
        (DIFF (IFTHEN D D) 2))
      2))
  (IFTHEN
    (IFTHEN C 4)
    (DIFF
      (IFTHEN
        (DIFF D (DIFF (IFTHEN D D) 2))
        D)
      2)))
```

Fig. 14. A sample of a complex, evolved strategy

hopes that this will increase the genetic program's probability of success. As is typical among this type of research, we are only interested in attaining 100% variety in the initial population; after that, duplicates are something that the genetic program naturally creates.

Maximum number of generations is a parameter that is generally set to keep the genetic program from running forever. We have no initial plan for the value of this parameter. Some very difficult problems require many generations (> 500) while simpler problems take far less time to solve (< 50). We will just have to see how effective and efficient the program is at solving this problem. There are a wide variety of settings that need values before the experiment is run; these are shown in Appendix B.

3.2 Fine-Tuning Parameters

The following parameters are not central to our investigation but may potentially have some impact on the performance of the evolutionary process. We will perform some cursory experiments to determine how the process performs at the different settings; in these experiments we will take the settings of the parameters in §3.1 as given. We will be looking at two performance measures: 1) **effectiveness**: the effect of the setting on the process's ability to find the best performing solutions, and 2) **efficiency**: the effect of the setting on the speed of the process in its search for the best performing solutions. We foresee measuring effectiveness in several different ways:

1. Average champion's ratio. Compare it with 1.0 to determine the equivalence of the final champion of the genetic program (from a certain scenario with the specific settings) to the standard strategy.
2. Average champion's ratio. Compare the score from one scenario with the score from others to determine what settings are preferred.
3. Percent of the time the champion's ratio is within 10% (for example) of the best champion's ratio across all experiments. This should reflect the robustness of the process with that setting.

We will measure efficiency in at least the following ways:

1. The average number of (simulated) weeks played in an experiment up until the time the champion is found.
2. The average number of games played in an experiment up until the time the champion is found.

Given the equal effectiveness of two different settings, we will prefer the one that enable more efficient searches. One of the findings that we expect from this research program is that we should be able to describe more completely the trade-off between effectiveness and efficiency in this problem.

We will use the above statistics to determine the effects of the following parameters on the effectiveness and efficiency of the genetic program.

Selection method As we discussed above in §2.3, the three possible means of selecting individuals for reproduction are fitness proportionate, rank, and tournament. We will also use the random selection method as a baseline for these methods.

Percentage of internal crossover points In an S-expression the internal points are the function nodes. If crossover is performed on two external (terminal) points, then this mimics the mutation operation on terminals. We want to encourage the creation of more complex trees so we will mostly investigate relatively large values for this parameter: 0.7, 0.8, 0.9, 0.95, and 1. Koza uses a value of 90% in his investigations [Koz00, p. 114].

Probability of crossover This is the percentage of the population on which crossover is performed. Koza uses the value of 0.90 [Koz00, p. 114]. For that portion of the population on which crossover is not performed, reproduction is performed (that is, those members are directly copied into the new population). The purpose of investigating this is to see what combination of crossover and reproduction makes the evolutionary process perform best. The possible values that we will investigate are 0.2, 0.5, 0.8, 0.9, 0.95, and 1.

Probability of mutation Koza generally does not use the mutation operation [Koz00, p. 114]. We hypothesize that the mutation operation might be useful for helping the genetic program avoid local minima in its search process. We will investigate possible mutation probabilities of 0.0, 0.01, 0.05, 0.1, and 0.25.

Since the allowed values for the last two parameters are dependent, we will have to investigate these two values in separate experiments. To look at the effects of the first three parmeters requires that a $4 \times 5 \times 6$ (120) completely randomized experimental design be employed in order to assess the impact of the above variables on the effectiveness and efficiency of the genetic program. We will use the settings indicated by the results of this experiment our investigation of the mutation parameter. In both experiments we will choose a moderately difficult problem for which an optimal (or heuristic) solution is already known. This will allow us to stop the evolutionary process when the solution is found or when the best value gets within some arbitrarily close range of this answer.

We will use these results in the following set of investigations.

3.3 Basic Investigation

The follow parameters are also not central to our investigation but may potentially significantly affect the performance of the evolutionary process (as opposed to the above parameters that we do not believe should have more than a marginal affect on the process's effectiveness though they might end up having a measurable impact on efficiency). We will use the settings of the

parameters in §3.1 as given and will use the values of the parameters that we determined in the experiments described in §3.2.

Number of populations (Φ) As the problem is described in §2, each role is assigned one population and *vice versa*. In those instances in which the parameters for each of the roles are the same, it would be computationally more efficient if all of the players came from one population. Further, it would be even more efficient if one population member were selected and then assigned to all four roles. We want to determine if this parameter's setting has an effect on the process's effectiveness. Possible values of 1 (one selection plays all the roles), 1 (one selection for each role), and 4 (one population per role).

Members per population (Λ_ϕ) One of the parameters that can have a significant effect on the process's effectiveness is the number of members within a population. Koza uses a population size of 500 about $\frac{2}{3}$ of the time [Koz00, p. 98]. This parameter is also one of the primary determinants of the process's efficiency. The values for this parameter that we are going to investigate are 50, 200, 500, and 2000.

Games per member per generation (M) If an agent in a role is playing in a value chain whose other players are being fulfilled by different agents, then we need to define a minimum number of games per generation that need to be played by each member. For comparability across different population sizes, possible values for this parameter are specified in terms of the percentage of other members that the member will play in each generation. The values that we are going to investigate are 5%, 10%, and 20%. This only applies to the last two "number of populations" choices above.

Turns per game (W) The number of turns per game has at least two important effects on the scores that players end up earning in the game. First, longer games help minimize the effects of the player's initial inventory position, focusing the scoring more tightly on the player's re-stocking strategy. Second, longer games allow for a more complete realization of the negative effects of a bad re-stocking strategy; if it has not become obvious by 35 weeks, then it almost certainly would become obvious by 100 weeks. The possible values that we are going to investigate are 35 and 100. (We chose 35 because that is the length of the MIT Beer Game, and we chose 100 because it seems suitably large.) In both cases we will not use 35 (or 100) *per se*; we will use a random number distributed around 35 (or 100) in order to keep the agents from over-adapting to the number of weeks.

Priming of initial population When an initial population is *primed* that means that relatively higher fitness agents are put into the population [Koz00, p. 94]. Koza proposes, and we agree, that it makes sense either to prime the whole population with relatively equally capable agents or to not prime at all; thus, possible values for this parameter are either "yes"

or "no." This brings up the obvious next question of how to perform this priming. We hypothesize that a useful approach would be to start with a high population (maybe 2000), run the simulation for one generation, choose the best 500, and then run the evolutionary process as normal with this smaller population. However, this is only a hypothesis. A full investigation of this parameter will have to wait until we understand more about this problem.

We are not going to be looking at determining the "best" value for each of these parameters. Here we simply want to gain insight into 1) the effect that these factors have on the effectiveness and efficiency of the search, and 2) the dependencies among these factors. We will use insight gathered from this process in the investigations outlined in the next section.

3.4 Initial Investigation into Information Value

The parameters discussed in this section form the heart of our investigation. We looked into the parmeters discussed in the previous sections because we wanted to ensure that the investigations into the parameters in this section were performed as efficiently and effectively as possible. Further, the parameters in this section can take many values and have many interdependencies so this will be a complex and time-consuming series of experiments.

Items in the function and terminal sets We have currently defined 21 functions: mathematical (7), logical (8), and informational (6). We can add the functions in each of these three sets to the function set either as a group or individually in order to see what effect they have. We have a smaller set of terminals — four plus the set of integers. Certainly one question relates to how small we can make the set of terminals and still enable the evolutionary process to proceed.

These are some further questions that we have: Which functions and terminals are used by higher-performing agents? We can investigate this question from two different perspectives. 1) One way is to manipulate the demand distribution and other parameters and observing which functions and terminals end up in the higher-performing agents. Do certain of the components appear in higher-performing agents? 2) Another way is to hold the functions and terminals fixed in three of the populations and manipulate the function and terminal sets. Do certain components lead to absolute dominance in a population? Another question relates to investigating whether or not certain fitness measures require that successful agents have certain terminals or functions.

In short, the questions in this section relate to examining questions related to valuing information.

Type of demand function This is one of the primary ways in which we can manipulate the difficulty of a scenario. Depending on how successful

the genetic program is on simpler problems for which an optimal solution is already known, we could run the evolutionary process with one of many values: standard MIT demand function, uniform distribution, Poisson distribution, cyclical variations, and non-stationary variations. Other parameters which can be used to make the problem more difficult: shipment lead time (determinstic or stochastic), information lead time (0, 1, 2). If the genetic program is successful at finding solutions for difficult problems, then we will compare the findings of this process with the findings of management science and information systems.

Inventory holding costs and penalty costs We will manipulate the values of these two parameters to determine if the genetic program is sophisticated enough to respond to this type of control. That is, will it be able to find low-inventory-level solutions when the first is relatively higher and high-inventory-level costs when the second is relatively higher.

Fitness measure As we discussed in §2.2, several different fitness measures are available for this problem. Three (based on A, Υ, and η) are appropriate for encouraging an agent to focus on its own costs; two others (based on T and Θ) encourage an agent to focus on the costs of the whole value chain. We want to investigate how much the different incentives actually affect the performance of the agents in the game in different scenarios. In order to address this problem, we will have to determine how to assign credit to members of a value chain that perform well. We have discussed using T and Θ, and it might be as simple as dividing the cost by the number of participants in the supply chain — but it might not.

Maximum depth overall The most effective way of controlling the complexity of the genetic programming problem is to limit the maximum depth of the tree representing the agent's re-stocking strategy. This places a limit on the number of different functions that the genetic program has to explore. We will manipulate this parameter as a means of exploring how agents can handle information overload. The possible values that we will look at are 6, 12, 17 (Koza's standard value), and 25.

Minimum depth of initial population This parameter provides a way of making the genetic programming process easier or harder. If the solution to one of the scenarios is simple, then removing all the simple structures from the initial population (by setting this parameter to 5) would make the genetic program have to discover the structure by itself, rather than having it be created in the initial population. Possible values that we will look at are 1 (Koza's standard value), 2, and 5.

3.5 Further Investigations into Valuing and Pricing Information

While the above certainly describes an ambitious research agenda, it by no means exhausts the possibilities. A much more complex supply chain scenario is one in which multiple players can play each level of the supply chain, and

a player must decide among them when placing an order. This would add a significant amount of difficulty to the genetic program.

To this point we have considered information and computational capabilities to be costless. For example, when a player uses the `sum` or `inv` functions, it cost him nothing. We propose that some additional cost should be placed on the agent for every additional piece of information that it uses in the calculation of its re-stocking strategy. Thus, the value of the information will be reflected not only in a positive sense — if the information is valuable (useful), then the re-stocking strategy will be more effective and the agent's possibility of reproducing into the next generation will go up — but in a negative sense as well — if the information is not valuable but is being used anyway, then the agent will have to bear the burden of the cost of collecting and using that information. We will change the raw fitness function for this set of investigations to reflect the cost of using information, and then we will see how this changes the genetic program's process.

The most interesting possibility lies in adding another type of competitor to this scenario. Above we discussed adding information cost to the player's fitness function. Here we propose that a population of information sellers be added to the scenario as a competitor for all the players in all of the roles. The fitness function of the information sellers will reflect its ability to sell information for as much as it can. The result of this experiment should be better insight into which information is most valuable to the agents in determining their re-stocking strategies.

4 Summary

This paper describes an ambitious research program that will use genetic programming to investigate re-stocking strategies in a multi-level supply chain. After describing the basics of the evolutionary process (§2), we describe our overall plan of investigation (§3). In this investigation we will be looking at questions related to the value of information, the usefulness of the genetic programming approach to this problem, and the supply chain management problem itself. We describe the design of an early experiment (§3.2) that we will perform to help determine some settings for the evolutionary scenario. We next discuss (§3.3) some further investigations into parameters of the process that will have a large effect on the efficiency and effectiveness of the process. We follow this up (§3.4) with a discussion of our proposed investigations into the value of information and computational capabilities, into the ability of the genetic program to respond to changes in fitness measures, and into the ability of the genetic program to handle larger search spaces. We finish the explanation of our research project (§3.5) with a discussion of alternative approaches for investigating the valuing and pricing information, the most exciting of which involves sellers of information co-evolving with the agents evolving in the supply chain game.

We have great hopes for what we can learn about valuing information, genetic programming, and supply chain management. Many researchers have addressed these problems separately, but we think much is to be gained by looking at them together. All that remains is to find out what that is.

A Terminals and Functions

A.1 Terminals

The following are all the terminals we used in our experiments:

1. `j` $\in$ set of integers
2. `I`: the amount of the good that the agent has in stock
3. `O`: the amount of the good that the agent has on order
4. `D`: the current demand for this agent
5. `C`: the current week number of the game

A.2 Functions

The following are the functions that can be used in the construction of an agent's strategy:

Mathematical

1. `(sum X Y)`: returns $X + Y$
2. `(diff X Y)`: returns $X - Y$
3. `(% X Y)`: returns `round(X Y)`, the result of rounding $\frac{X}{Y}$ to the nearest integer; if $Y = 0$, returns 100000.
4. `(rmndr X Y)`: returns rem(XY); e.g., `rem(13,5)=3`
5. `(mult X Y)`: returns $X * Y$
6. `(min X Y)`: returns the minimum of `X` and `Y`
7. `(max X Y)`: returns the maximum of `X` and `Y`

Logical

1. `(ifThen X Y)`: if `X` does not evaluate to `0`, returns value of `Y`, otherwise returns `0`
2. `(ifThenElse X Y Z)`: if `X` does not evaluate to `0`, returns value of `Y`, otherwise returns `Z`
3. `(not X)`: if `X` does not evaluate to `0`, returns `0`, otherwise returns `1`
4. `(and X Y)`: if `X` and `Y` are true, returns `1`, otherwise returns `0`
5. `(or X Y)`: if `X` or `Y` are true, returns `1`, otherwise returns `0`
6. `(gt X Y)`: if $X > Y$, returns `1`, otherwise returns `0`
7. `(lt X Y)`: if $X < Y$, returns `1`, otherwise returns `0`
8. `(equal X Y)`: if $X = Y$, returns `1`, otherwise returns `0`

Informational

1. `(MYDEM Y)`: the player's own demand `Y` weeks ago. The function uses the modulo function on `Y`; the demand is actually calculated for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).
2. `(MYINV Y)`: the player's own inventory `Y` weeks ago. The function uses the modulo function on `Y`; the inventory is actually retrieved for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).
3. `(MYORD Y)`: the player's order that it placed `Y` weeks ago. The function uses the modulo function on `Y`; the order is actually retrieved for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).
4. `(DEM X Y)`: player `X`'s demand `Y` weeks ago. The function uses the modulo function on `X` to map from integers to game players. For example, if `mod(X, 4)=3`, then the *retailer* would be chosen. The function also uses the modulo function on `Y`; the demand is actually retrieved for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).
5. `(INV X Y)`: player `X`'s inventory `Y` weeks ago. The function uses the modulo function on `X` to map from integers to game players. For example, if `mod(X, 4)=3`, then the *retailer* would be chosen. The function also uses the modulo function on `Y`; the inventory is actually retrieved for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).
6. `(ORD X Y)`: player `X`'s order that it placed `Y` weeks ago. The function uses the modulo function on `X` to map from integers to game players. For example, if `mod(X, 4)=3`, then the *retailer* would be chosen. The function also uses the modulo function on `Y`; the order is actually retrieved for `mod(Y, W)` weeks ago (where `W` is the number of weeks in a game).

B Settings for a Scenario

The following list shows the range of values that are set up before a scenario is run.

1. game-type: ONE-POP-FOR-ALL
2. num-of-populations: 1
3. members-per-population: 200
4. num-of-roles: 4
5. players-per-role: 1
6. maximum-number-of-generations: 300
7. generative-method-for-initial-population: RAMPED-HALF-AND-HALF
8. how-to-measure-player-performance-in-a-generation: COSTS-FOR-THE-PLAYER
9. fitness-measure: NORMALIZED
10. selection-method: TOURNAMENT
11. internal-crossover-points: 0.9
12. max-depth-for-individuals-after-crossover: 17

13. min-depth-for-new-individual: 2
14. max-depth-for-new-individual: 6
15. probability-of-crossover: 0.5
16. probability-of-mutation: 0.02
17. probability-of-permutation: 0
18. probability-of-editing: 0
19. encapsulation: 0
20. probability-of-decimation: 0
21. prime-percent: 0.25
22. prime-type: SCOTT-PRIME
23. seed-strategy: (D D D D)
24. name-of-terminal-set: STANDARD
25. name-of-function-set: STANDARD
26. type-of-demand: UNIFORM-DEMAND
27. minimum-uniform-demand: 5
28. maximum-uniform-demand: 95
29. demand-cycle: none
30. weekly-increase: 0
31. games-per-member: 5
32. average-weeks-per-game: 75
33. inventory-holding-costs: 1
34. penalty-cost: 5
35. beginning-inventory: 100
36. backorder-amt: 0
37. amt-received-in-last-shipment: 50

C Computing-Related Information

These experiments were run on machines with Intel or Intel-compatible chips. Experiments with 200 members, 300 generations, 1 population, and 5 games per member (the factors that most significantly affected the time these experiments took) might take, for example, 2.4 hours to complete the 8.3 million simulated weeks in the experiment.

We wrote the program that runs the simulations for this project using GNU CLISP [Fre04]. During the course of execution, this program generates a trace file that captures the basics of the status of the genetic program. The trace file it generated would end up being around 30MB. After the scenario is done, a Python program reads the trace file and generates a LaTeX file; the graphics it contains are generated using the PSTricks macros. After this file is generated, the `dvipdf` program (part of the `tetex` distribution) is used to generate an associated PDF file. In the above scenario this file would be around one thousand pages and around 1.2MB; the first fifteen pages of this document generally show relevant graphics and summary statistics while the remaining 900+ pages simply print out the best player from each generation. Clearly, the more generations, the longer this part of the document will be.

Endnote

The inspiration for this paper comes from reading Kimbrough, Wu, and Zhong's paper from FMEC2000[1] as well as interesting conversations held after the paper was presented. We have also benefitted from several long discussions concerning GAs and GPs with Michael Gordon.

References

[Che99] Fangrou Chen, *Decentralized supply chains subject to information delays*, Management Science **45** (1999), no. 8, 1076–1090.

[Fre04] Free Software Foundation, Inc., *GNU CLISP — an ANSI Common Lisp*, http://clisp.cons.org/, June 2004.

[Hol92] John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, 1992.

[Koz00] John R. Koza, *Genetic programming: On the programming of computers by means of natural selection*, A Bradford Book/The MIT Press, 1992 (7th printing, 2000), ISBN 0-262-11170-5.

[KWZ02] Steven O. Kimbrough, D. J. Wu, and Fang Zhong, *Computers play the beer game: Can artificial agents manage supply chains?*, Decision Support Systems **33** (2002), no. 3, 323–333.

[MD04] Scott A. Moore and Kurt Demaagd, *Beer game genetic program*, http://sourceforge.net/projects/beergame/, 2004.

[Ste89] John Sterman, *Modeling managerial behavior misperceptions of feedback in a dynamic decision making experiment*, Management Science **35** (1989), no. 3, 321–339.

[Ste04] John D. Sterman, *Teaching takes off: Flight simulators for management education*, http://web.mit.edu/jsterman/www/SDG/beergame.html, 2004.

[1] Editors' note: subsequently published as [KWZ02].

Multi-Agent Simulation of Financial Markets

Olga Streltchenko[1], Yelena Yesha[2], and Timothy Finin[3]

[1] Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA, streltch@cs.umbc.edu
[2] Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA, yeyesha@cs.umbc.edu
[3] Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA, finin@cs.umbc.edu

Abstract. This paper discusses the principal reasons for, and prospective opportunities of, simulating financial markets using an architecture based on artificial agents. The paper then discusses in detail the design and architecture of a simulator for financial markets. The Gaia methodology was employed in the development of MAFiMSi (Multi-Agent Finanacial Market Simulator), a general-purpose finacial market simulator of a dealer-type market. MAFiMSi is implemented as a library of C++ classes that currently support a stand-alone market simulation.

1 Introduction

Simulation of financial markets is a new fast growing research area with two primary motivations. The first is the need to provide a development testbed for the ever increasing automation of financial markets. The second is the inability of traditional computational mathematics to predict market patterns that result from the choices made by interacting investors in a market.

Section 2 surveys the current state of financial market automation. Section 3 discusses the importance of simulation to help understand the patterns that arise from different investment strategies; it briefly surveys the literature and identifies some open problems, including the design of a general-purpose financial market simulator.

The design of a multi-agent simulator of a financial market is the subject of Section 4. It is a challenging task due to the operational complexity and computationally costly decision support. We discuss an approach in which the complexity of the financial market functionality is decomposed into relatively simple tasks and processes. In general, we separate the transactional and decision-support intelligence of the market agents. Futher, market entities are singled out and defined as software objects; the interaction protocols are specified; and the simulator architecture is presented.

We conclude with the discussion of possible applications of the simulator, stressing it extensibility to handle other marketplaces, such as emerging markets, energy and bandwidth markets, etc.

2 Automation of Modern Financial Markets

We define market automation as the execution of trades by software agents based on goals specified by human agents.

Modern financial market professionals recognize the benefits of financial market globalization, worldwide trading through electronic interconnectivity, and around-the-clock market accessibility, as a means to increase liquidity and market efficiency. Automation is seen as a way to achieve these ends. Frank Zarb, chairman and CEO of NASD, formulated a vision of digital, global, continuously available security trading with real-time quotation and order execution systems accessible worldwide over the Internet via a number of computing devices in his speech to the National Press Club in 1999 [Zar99]. It forecasts rendering physical trading floors obsolete and completely replacing them by electronic transactions. Complete automation is necessary to support 7/24 availability.

This vision of financial market automation is not isolated. It is a part of a wider phenomenon of globalization and development of digital economy. The technology that can turn this vision into reality has arrived. The trend to automation is supported by developments in electronic commerce, agent technology, and achievements in mathematical and computational finance.

This section gives a brief introduction into the operations of financial markets and survey the current state of financial market automation. Electronic Communication Networks (ECNs) are of special interest here due to their pioneering role in automation of securities trading.

2.1 Securities Markets

Modern financial markets deal in standardized obligations in place of goods and commodities. In the current trading paradigm, the actual order execution, or securities exchange, is separated from an investor by several levels of intermediaries (see Figure 1). In general, a hierarchical structure is a characteristic feature of modern securities trading. Access to the actual financial market is open only to authorized brokerage houses. Institutional and retail customers contact a broker to place their orders. Once an order is initiated by an investor (placed with a broker) it goes through three distinct stages: order routing, execution, and clearing and settlement. Routing involves communicating, possibly through a number of intermediaries, the details of an order from a broker with whom the order has been placed to a market agent, human or software, responsible for order execution. Execution involves agreement to exchange securities, while clearing and settlement commits the transaction by

checking availability of the resources committed by both sides and exchanging the securities.

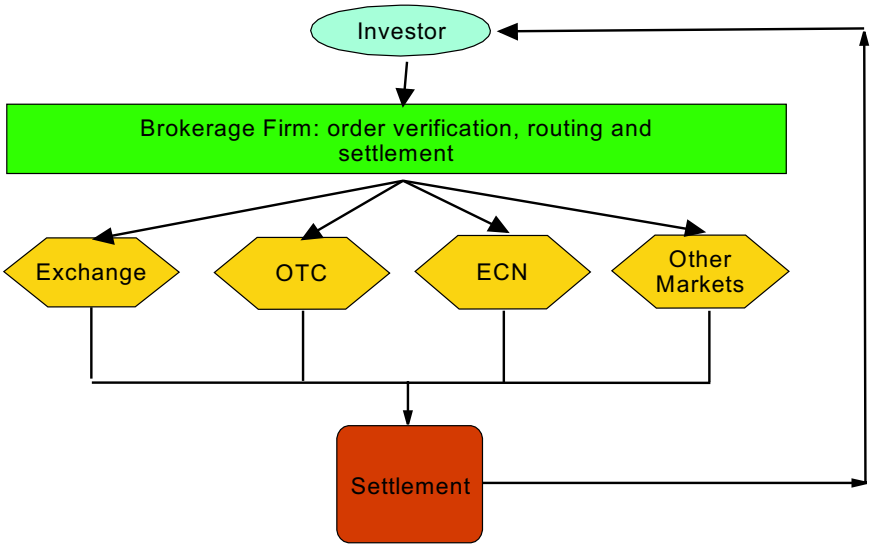


Fig. 1. Current trading model; adopted from [FSW00]

Not only the objects of exchange, but the exchange protocols have become standard. Exchange protocols are designed to achieve a price for a security that the trading community agrees on, i.e., is willing to trade; this procedure is called a *price discovery mechanism*. There are two major types of markets defined with respect to the price discovery mechanisms that they implement: the *dealer market* and the *auction market*, with a number of hybrid types between these two extremes. In an auction market a market specialist acts as a facilitator and does not own traded securities. Trading proceeds as a continuous double auction with the market specialist acting as an auctioneer assisting the sides in matching their offers and arriving at a mutually agreed on price for the trade. There is only one market specialist for a particular security. In a dealer market a market specialist owns the securities being traded. A dealer posts his/her ask and bid prices, and all the transactions occur between him/her and the investor (more precisely, an agent acting on behalf of the investor). Dealer markets allow several competitive dealers trade in a particular security.

Security trading occurs at exchanges, over-the-counter markets (OTCs), and electronic communication networks (ECNs). An exchange, such as New York Stock Exchange (NYSE) [PBR95], is an example of a hybrid market with respect to its price discovery mechanism. It operates as an auction mar-

ket most of the time, with the market specialist occasionally acting as a dealer when the market conditions require so. This might happen when there is a considerable disparity between supply and demand that threatens to dramatically swing the prices, and a stabilizing intervention of the market maker is desirable to the investor community to prevent trading from stalling. Over-the-counter markets, such as NASDAQ, are dealer markets. ECNs are neither auction nor dealer markets. In fact, they are not considered to be true markets since they lack a price discovery mechanism. They will be discussed in detail below.

A large degree of standardization exhibited by modern financial markets allowed to automate certain market procedures. We will now proceed to discuss the current state of financial market automation.

2.2 Current State of Financial Market Automation

As was stated above, there are several levels of intermediaries to a financial transaction spanning order routing, execution and settlement. The efficiency of such a pyramid depends on the speed and quality of communications between the parties as well as streamlining of clearing and settlement for the trade.

Advances in clearing and settlement automation were made in in the 1970s [PBR95]. The standardized obligations traded on financial markets readily yielded to software representation due to their abstract nature; the security ownership information is now almost completely relegated to digital information systems.

Throughout the century financial markets made heavy use of every advance in the communications technology. Currently, information exchange is almost fully automated. Electronic quotation systems came into existence in the early 1970s [PBR95]. These display price quotes as well as post-trade information, such as transaction volumes, etc.

Order routing became automated with the development of order routing systems like SuperDOT (Super Designated Order Turnaround) at the NYSE, which significantly increased trading throughput. Currently, all exchanges and most OTCs receive most of their order volume through automated order routing systems.

Order execution requires the most human involvement. It has been automated for small-sized orders by some exchanges and OTCs. Such automated order execution systems handle both market and limit orders. As a rule, there is an upper bound on the transaction volume for such a system. This number varies for a particular security depending on how frequently the security is traded, i.e., depending on its liquidity. An example of an automated order execution system is SOES of NASDAQ which became ready for use in 1984 and allowed transaction volume up to 1000 shares.

Extending market activities into after-hours and competition from ECNs (see below) has forced further sophistication of automated execution systems.

For example, in the 1990s SOES was enhanced by SelectNet which has certain negotiation capabilities. That allowed the system to implement basic price discovery, handle disproportionate orders, and, therefore, to be able to execute transactions of higher volume.

2.3 Electronic Communication Networks

ECNs were designed to compete with existing market makers (specialist) on the financial markets, particularly NASDAQ. The first ECN to become operational was Instinet, which opened in 1969. Currently the number of competitive ECNs is 9.

ECNs are completely automatic: no human market maker is involved in trading. They emulate the operations of an auction-type exchange by electronically matching buyers' and sellers' orders. However, the original matching procedure was very basic: orders to sell and orders to buy were gathered from the members and searched for matches in volume and price. If a match was found, a transaction went through. Otherwise, nothing happened. Such a procedure does not guarantee order execution; it remains a passive matchmaker and has to avoid disproportionate client portfolios. Since there are no price negotiation capabilities, it does not lead to price discovery, and, therefore, cannot be considered an electronic market. Traditional exchanges became sources of price information for ECN members.

Despite the above mentioned drawbacks, ECN market share has been steadily growing. This is due to their after-hour trading capacities and low transaction costs - the benefits of complete automation.

The early example of automated order execution by the ECNs stimulated the development of automated order execution systems within the traditional markets. ECNs responded to the competition by extending their functionality to include order negotiation (price discovery). Once full market functionality is achieved, an ECN can apply for a change of status and become an exchange. Archipelago, Island, and NextTrade have done so.

2.4 Further Examples of Financial Market Automation

The major factor currently driving financial market automation is the Internet. Its influence on securities trading has been dramatic; investing is now as easy and accessible as playing an internet game. Its effect on the investor community is now under scrutiny by the academic community; for an early attempt to discuss the issue, and, more generally, the effects of financial market automation, see [Var98].

The emergence of online trading was marked by the appearance of purely electronic brokerages, along with traditional brokerages opening online trading sites. The Internet has gone beyond just being another channel for order placing; opportunities for faster and more comprehensive research into an individual company or security performance, visualization and analysis tools

for processing historic data offered on online brokerage sites, 7/24 accessibility are just a few services offered by online brokerages. Note that 7/24 availability of an online broker does not directly translate into 7/24 availability of financial markets. Orders can be placed with a broker continuously, but they will be executed according to the trading schedule of a chosen financial market. A number of issues concerning online brokerages were studied in a Securities and Exchange commission special study report [SEC99].

Advances in automation have not so far reduced the number of intermediaries to a transaction. For the problems that arise as a result of the hierarchical structure, such as order internalization, and their effect on overall market efficiency, see [FSW00]. The paper assumes an optimistic attitude and offers a direct trading model (Figure 2), in which disintermediation is facilitated by completely automated order routing.

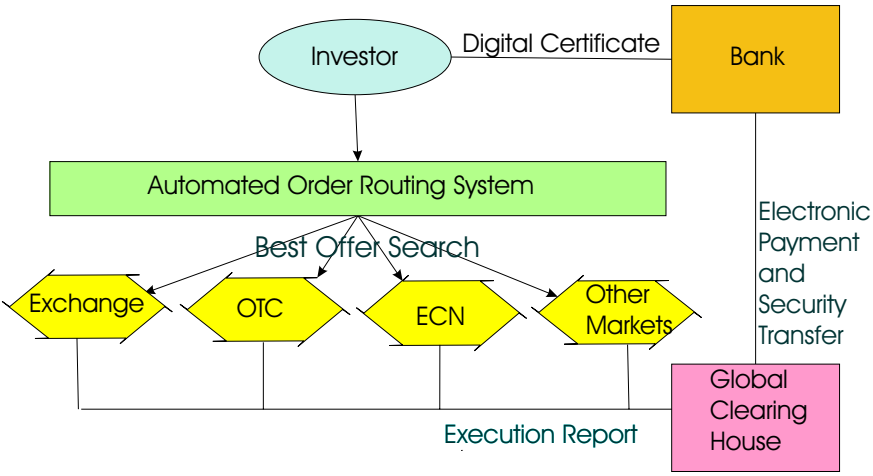


Fig. 2. Direct trading model; adopted from [FSW00]

Another maturing aspect of financial market automation is automated decision support on the part of the investors. A large fraction of the investor community, e.g., all institutional investors, relies on mathematical / computational modeling of the market to support decisions concerning their portfolios. In most cases, the computational results are verified by a human agent before corresponding orders are made. A notable exception to this procedure is program trading. Currently program trading employs fairly straightforward algorithms to support decisions about portfolio re-balancing. They implement what is called technical trading: searching for particular trends in the price movements of a chosen security(ies) and responding in a predefined manner once the trends have been observed. In general, the performance of techni-

cal trading is hotly debated by the financial community. In one particular instance, unsupervised technical trading—program trading—had a devastating effect on the stock market; during the 1987 stock market crash program traders picked up, followed and amplified the trend to sell [Var98]. The lesson of 1987 suggests the importance of the empirical study of large systems populated by automated decision-makers of various degrees of intelligence.

However, the importance of automated decision support for investors is almost self-evident. A financial market is a highly dynamic environment where the ability to act upon the latest price quotes is crucial: if a price moves away from the quote used to make a decision significantly while the decision is being made, the investor cannot be sure that his/her assessment of his/her position will hold in a significantly changed environment. The same argument applies to a market maker who needs to update price quotes of the securities that he/she is responsible for to maintain a dynamic balance between supply and demand. The speed of response to market signals is not the only incentive: automation is a precondition to global continuous, 7/24 trading.

3 Simulation of Financial Markets

The complexity of financial markets defies traditional mathematical and computational analysis [Rus96]. Since many of the questions we would like to ask about financial markets are not amenable to theoretical analysis, experimental analysis suggests itself. The idea of experimental study originates in physical sciences, where a controlled experiment involves repeatability and parameter isolation. A financial market permits neither of these. (For example, we can not hold inflation and interest rates steady, as we experiment with a variety of pricing techniques.)

The impossibility of conducting controlled experiments has been identified as one of the major hindrances for transition of empirical finance into an axiomatic theory [FJ97]. Multi-agent simulation of financial markets seeks to address this problem by providing the conditions for a controlled experiment, and thus allowing us to isolate cause and effect relationships in the market. The use of multi-agent simulation, therefore, may help greatly advance the theoretical developments of finance theory.

For instance, multi-agent simulation can aid in the understanding of derivative securities pricing for which intuition often takes the place of exact science among the practitioners [Par97]. This is especially true for energy and bandwidth contracts, since storability of the underlying - a fundamental assumption of traditional derivatives pricing techniques - is not a characteristic of these markets.

In traditional mathematical finance, all market participants are modeled the same way, with each having equal powers, and being subject to the same constraints. This is an idealized setting. For example, airline companies and

oil refineries both come to the market to trade in fuel, but they do so with fundamentally different perspectives, one of them being required to sell, and the other required to buy. Speculators have a perspective different from either. One can identify an arbitrary number of individual investor profiles in a single market. A theoretical study of such a model is virtually impossible. The strength of multi-agent approach is the ability to (experimentally) study large heterogeneous populations.

In this section we survey the work of three research communities whose work, in our opinion, comprises the background for further advances in the field. We distinguish two main branches in this area of research. One concerns itself with setting the environment, e.g., simulator and agent architecture, functionality and implementation. The other one concentrates on specific application domains, special attention being given to agents' intelligence (decision-support algorithms) and environment modeling. Most research groups have not drawn this distinction, and the projects we survey below have made contributions to both branches.

3.1 Trading Agent Platforms

Motivated by the idea of facilitating electronic purchases over the Internet and reducing human involvement in corresponding search and, in some cases, negotiation, several research groups developed trading agent platforms. Several such systems are surveyed in [MGM99]. The features that received special consideration in such systems are

- search for the most suitable product or a set of alternative products;
- search for the best merchant or a set of merchants for further negotiation; and
- negotiation itself with a purchasing decision at the end of the process.

Not all of these features were necessarily implemented; most of the agent systems offered Web-search capabilities while leaving negotiation and/or decision-making to the user.

Several agent platforms, such as Kasbah, e-Mediator, and AuctionBot, implemented an auction-type interactions or one-on-one price negotiation and offered a number of bidding strategies to be chosen by the user prior to negotiation. Such systems are of interest to us since they actually emulate a certain marketplace as well as well as supply basic regulatory and administrative infrastructure.

eMediator, an electronic commerce server from the Multi-Agent System Research Group at Washington University described by Sandholm, is arguably the most advanced of this family [San99]. It features eAuctionHouse that implements a choice of auction types, strategic (price-quantity graph) bidding and combinatorial bidding (bidding on a set of products), and creation of personalized Java agents that perform trading on the server; eCommitter, a decision support engine for leveled commitment contract optimizer;

and eExchangeHouse, an exchange planner. A number of other features are under development. The research that developed around the server mostly concerns semi-binding commitment (level commitment), agent strategic behavior in an auction-type environment, namely coalition formation, and combinatorial bidding winner determination. We recommend that the interested reader peruse a collection of publications by the Multi-Agent System Research group at `http://www.cs.wustl.edu/~mas/`.

The systems discussed above facilitate product search on the Internet while providing the user with a varying degree of negotiation and/or bidding support. Mostly they rely on fairly simple interactions and decision rules. Those building a financial or economic system simulator can benefit from their advances in agent platform development, agent architecture, distribution, and other system related issues as well as the decision support provided for bidding and negotiations. However, such systems do not provide the infrastructure and decision support specific to the financial market domain.

3.2 Simulation of Simple Economies

Auction as a price discovery mechanism has received significant attention in the experimental research of multi-agent interactions. Part of the reason for this is that individual agents rely on fairly simple decision rules in such an environment. However, the apparent simplicity of such systems may be deceptive.

Evidence of the emergent complexity in systems employing simple pricing techniques is presented in a series of publications on multi-agent simulation of simple economies by the Information Economies research group at IBM's T. J. Watson Research Center.[1] It is important to note that oversimplification of decision support in a multi-agent system can lead to disastrous consequences for the economy, such as price wars and stagnation of trading, examples of which are give in Tesauro et al. [TK98] and Brooks et al. [BDD00]. This phenomenon is not confined to experimental systems; recall the example of the stock market crash of 1987 given above. Due to such possible outcomes, Kephart, Hanson and Greenwald in [KHG00] stress the importance of multi-agent simulation as a testbed for novel decision support algorithms before implementing them in a real-world system.

The agent decision support algorithms chosen by this research community are variations on Q-learning, an instance of a wider class of reinforcement learning algorithms. Kephart et al. [KHG00] overview a number of experiments with an information economy populated by heterogeneous agents, some of which employ Q-learning. The model of the information economy presented in [KHG00] assumes a dynamic posted pricing paradigm, that is a model in which buyers do not negotiate posted prices while sellers update their postings at will. The economy is similar to a commodities market

[1] See, for example, [KHL+98b] and [KHL+98a].

populated by rational self-interested agents optimizing their utility function via one of a number of algorithms ranging from a simple incremental price increase(decrease) algorithm to foresight-based Q-learning.

While financial markets are not explicitly considered here, the multi-agent systems under consideration possess the price discovery mechanisms present in financial markets. Insights into the behavior of large systems presented by this community can help to better set up the experimental environment for financial market simulation.

3.3 Financial Market Simulation

While the above examples use the price discovery mechanisms similar to those present in financial markets, they do not explore the financial domain explicitly. We now proceed to discuss simulations specific to financial markets: the Santa Fe Artificial Stock Market and AGEDASI TOF, A GEnetic-algorithm Double Auction SImulator of TOkyo Foreign exchange market.

Several early efforts in the area of simulated financial markets are surveyed by LeBaron in [LeB00]. The paper recognizes the strength of an agent-based approach in understanding the dynamics of interaction among heterogeneous agents and agents learning from the environment. The survey concentrates on the concrete microeconomic (market) models and learning techniques, predominantly genetic algorithms, used in the surveyed work.

One of the most prolific research efforts in the area is the Santa Fe Artificial Stock Market, surveyed in [LeB00].[2] This line of research, along with further experiments on the same platform,[3] studies success, in a game-theoretic sense, of technical traders relying on a variety of learning techniques borrowed from machine learning to improve their forecasting ability and gain a competitive advantage over other market participants.

The experience of the Santa Fe Artificial Stock Market and a number of several other simulation efforts are summarized in [LeB01a]. The paper offers a classification of design issues that a builder of a financial market simulator may face. The first category, agents, discusses the types of market participant agents with respect to their intelligence. The stratification of agents on the bases of their intelligence parallels the classical AI classification offered by Russell and Norvig [RN95]. These agents are assumed to be price-takers, or regular investors, since the price setting issues are treated separately, in the next category, called trading. Trading covers both the trading protocol and the determination of asset prices. It differentiates between simulating the price movements and replicating market infrastructure with an appropriate price discovery mechanism. The former approach does not call for a market specialist agent while the latter does. The next design issue concerns

[2] Its original design and experiments are presented in [PAHL94], [AHL+97] and [LAP99]. For a brief digest of the model also see [Tes02].

[3] See, for example, [JPB99]

the securities themselves. The author notes the tendency of simplification in modeling the traded assets and discusses the difficulties associated with incorporating company fundamentals (earnings, dept structure, market share, product quality, etc) into the simulated securities. The number of securities in a simulation, and the related issue of diversification, are mentioned. Next, the issue of market evolution, or market agent learning, is tackled. Learning based on genetic algorithms is considered as a means of emulating dynamics and growing the expertise of the market participants. The necessity of model validation and possible approaches to it are discussed under the category of benchmarks/calibration. The category of time groups together several issues: agent memory for learning algorithms, lag in response to new information arriving to the market, and synchroneity.

[LeB01a] fails to discuss simulation time horizon. The issue of simulation horizon is important when modeling a community of investors with heterogeneous goals, like speculators, who invest for immediate reward, and long-term investors, who invest for future income. A financial market is populated by both types, and their interactions are important for maintaining dynamic balance. A simulation with a short horizon will concentrate on observing behavior of more active traders while a longer horizon will give insights into long-term investors' behavior.

LeBaron devotes a separate paper to the in-depth treatment of the issue of agent memory for learning algorithms in a financial market simulation and the influence of this parameter on the overall dynamics of the system [LeB01b].

In a series of papers by Izumi et al.,[4] multi-agent simulation was employed to study price dynamics in a foreign exchange market. The model received a name of AGEDASI TOF, which stands for A GEnetic-algorithm Double Auction SImulator of TOkyo Foreign exchange market. In the model a community of dealers derive their price quotes from quantifying information from various news sources through a system of weights assigned to each source. Dealer's success is determined based on the transaction volume, and a genetic algorithm procedure was used to adjust the weights for each agent. These simulations exhibited some interesting patterns, such as formation of dealers' opinion trends and clusters of agents' strategies.

Current effort in the area of multi-agent simulation of financial markets is heavily biased toward machine learning techniques, primarily genetic algorithms, to provide agents' decision support, as can be seen in the work of the two research communities reviewed above. The study is thus limited to technical trading, - the area that traditionally relies on the use of genetic algorithms. The vast majority of the investment community does not rely on technical trading; optimization in conjunction with risk management is the technique of choice for large (institutional) investors. Hedging and risk management are not directly addressed in technical trading. The genetic algorithm approach, or the survival of the fittest, tends to favor a winner with

[4] See, for example, [IU99a] and [IU99b].

the most money. Thus, risk-averse investors do not win in such a society, and get "weeded out" by the algorithm. Their success as hedges is not recognized by the procedure. Therefore, an important and large fraction of the investor community is not properly modeled by the genetic algorithm setting.

3.4 Concluding Remarks

Existing research concentrates on a series of individual simulation problems. There has not yet been a unified study of the common properties of financial market simulations. The focus of attention has been agent's intelligence. AI techniques clearly dominate the landscape, while traditional mathematical models of market agent decision support, such as (stochastic) optimization, (stochastic) differential equations, etc., have not been implemented for such systems. As was pointed out above, AI research operates under a heavy game-theoretic perception of success: finding a winner with the most wealth. This approach ignores the main concern of mainstream investors: risk management and hedging. In our opinion, multi-agent simulation of financial markets should

- elaborate its criteria of agent success and put more stress on risk management;
- broaden the choice of decision support techniques to include traditional portfolio management and security pricing techniques developed by the computational finance community;

Financial markets are highly regulated and standardized in their operation. While the motivation behind individual investors' actions can be extremely diverse, all market participants are subject to a fixed set of protocols that regulate securities exchange. These considerations suggest the necessity of a unified approach to formal modeling of financial market infrastructure - the issue that we proceed to discuss below.

4 A Multi-Agent Environment for Financial Market Simulation

This section addresses one of the open problem identified above, and offers a methodology for the analysis and design of a general-purpose financial market simulator. Our exposition builds on previous work. Although that work [SNY01] presents a methodology for a particular application, namely a derivatives market simulation, here we apply the idea developed in [SNY01] to a general-purpose financial market simulator.

The methodology to be presented separates the transactional and decision-making components of the simulated environment. This frees a researcher from the necessity of implementing the underlying market infrastructure, while allowing him or her either to choose among the available options for

agent decision support or to supply new algorithms. This also provides for independent developments in the areas of market functionality and decision support.

The methodology presented below models a generic dealer market. While a number of features we judged to be essential for a valid financial market simulation are recreated within the environment, certain simplifications were made to reduce its computational complexity. The simulator offers a universal quotation system, a unique (for each instance of the simulator) money-market, a security registration facility, order posting and trade clearing. The underlying infrastructure supports efficient and correct message flow among the trading agents.

Financial market participants are highly heterogeneous with respect to their ability to quote prices, engage in short-sales, borrow money, and many other respects. The model currently distinguishes one type of a privileged market participant, the *broker*, or *market specialist*, who has the unique power to quote a price, and who provides liquidity to the market. Other market participants are modeled after regular investors and are assumed to have no such power and to be price-takers. This basic division can be enhanced to accommodate several layers of market specialists and intermediaries with a variety of market powers. However, such fine stratification is not necessary for most basic applications, and, therefore, was not considered for the current simulator. Further we will refer to market agents as investors or brokers depending on their privileges with respect to quoting prices.

Brokers communicate their prices to the investor community via the quotation system. Investors convey their orders to chosen brokers, and they trade. However simple this scenario may sound, it comprises a lot of intricacies that have to be unraveled on the implementation level. Integrity of the exchanged information must be guaranteed, as well as timely delivery of the messages; trades must be appropriately cleared and monitored; investors' portfolios must be maintained, etc. In order to correctly implement the market operations the overall complexity of interactions and market functionality is decomposed into relatively simple tasks and processes. We further present a number of steps that achieve this goal. The decomposition is done according to *Gaia methodology* [WJK00], [ZJW00]. To facilitate our discussion we proceed with a brief review of Gaia methodology.

4.1 Gaia Methodology

Gaia methodology [WJK00] is a body of high-level software engineering techniques particularly suitable for hierarchical systems of heterogeneous agents which make use of significant computational resources. While offering a means of detailed analysis of the target application, it does not presuppose any particular implementation platform. It does not impose any checks on the overall system complexity or agent intelligence, but the organization structure of the system and the agent properties are assumed to be fixed during run-time.

Gaia recognizes *abstract* and *concrete* entities. Abstract entities are used during the analysis stage to conceptualize the system, while concrete entities are used within the design process and typically have direct counterparts in the run-time system.

The topmost abstract entity in the Gaia concept hierarchy is the system. Gaia views the system (organization) as a collection of roles that interact with each other. Each role is defined by its *responsibilities*, *permissions*, *activities*, and *protocols*. Responsibilities determine role functionality. In order to realize its responsibilities, a role possesses a set of permissions or rights associated with the role. Access to information sources, like information generation, modification or reading, is under the purview of permissions. Computations associated with a role are called activities. Roles can interact according to a set of protocols. The dependencies and various relations among the roles are captured by the *interaction model*. The interaction model is essentially a directed graph with roles as nodes.

Once the analysis produces fully elaborated roles and interaction models, the goal of the design stage is to transform the analysis models (abstract entities) into concrete entities having implementation counterparts. The *agent* model defines the agent types present in the system. The *service* model defines the actions performed by agents. The *acquaintance* model represents the communication channels among the agents.

The agent model can combine several closely related roles as a single agent type for efficiency. It is convenient to think of an agent type model as a tree with the roles being leaf nodes.

The service model expresses agent functionality. The services are derived from the protocols, activities and responsibilities of the individual roles combined into an agent type.

The acquaintance model is a directional graph with its arcs corresponding to communication pathways.

Below we describe analysis and design of a multi-agent environment simulating a financial market based on the guidelines of Gaia methodology. As we proceed, we will offer further explanation of the methodology when necessary.

4.2 Analysis

The analysis of the system aims to decompose the system's functionality into a number of atomic functions or actions. These atomic actions are segregated and attributed to the entities that perform them. This leads to delineation of the roles and results in a roles model that describes the system.

Our intention is to separate transaction processing from decision making. Therefore, each role will be responsible for either of them but not both. This distinction ultimately propagates to the concrete implementation of a financial market simulator described below.

Roles Model We introduce the following roles: Market, Investor, Broker, Investor Decision Support, and Broker Decision Support.

Orderly trading is impossible without a regulatory and administrative authority which provides the universal quotation system, security regulation, trade clearing, etc. The *Market* (M) role is responsible for providing market participants with a centralized administrative and regulatory environment. Its basic responsibilities include

- registering admissible securities upon requests from the brokers and maintaining a list of registered securities (securities available for exchange);
- maintaining a centralized offer posting service and posting new offers for registered securities originating from brokers.

Additionally, if a particular simulated environment poses certain security concerns (as may happen in a distributed simulation, for instance), the Market role may be responsible to act as a clearing house, an arbitrating authority, as well as perform some system functions as synchronization, quality-of-service assurance, etc.

As has been previously outlined, brokers establish prices and provide liquidity to the market. Investors are price-takers. All the trades have to be registered with the market (in order to verify the authenticity of the exchanged securities). Only brokers have write access to the quotation system, and, therefore, only they can be identified as potential counterparts to a trade. Thus, the investors trade solely with the brokers and are virtually transparent to each other.

The *Investor* role is responsible for

- maintaining a portfolio of currently held securities;
- obtaining information about desired changes in the current portfolio;
- obtaining appropriate price quotes along with the quoting broker information;
- choosing suitable Brokers based on their price quotes and serving them with corresponding market orders;
- maintaining a list of unacknowledged (open) orders.

Its responsibilities may also include serving limit orders to the brokers. The protocols that support these responsibilities are as follows. On behalf of the *Investor Decision Support role*, the Investor's decision-making counterpart (described below), the Investor queries the Market, namely its quotation system, to obtain security offer information and choose the best offer depending on whether it needs to buy or sell; forwards this information to the Investor Decision Support role; and receives desired portfolio information in reply. The Investor then makes provisional changes to its portfolio and initiates corresponding transactions - forwards orders for each individual security to appropriate brokers. Until an order is fulfilled, the Investor maintains a list of open orders. Once an order has been fulfilled by a broker, the Investor commits the portfolio update by removing the order from its list of open orders,

and acknowledges receipt of the order. Investor's read permissions open read access to the price quotation system while its read/write permissions allow portfolio and order manipulation.

The *Broker* role extends the Investor role with additional responsibilities which include

- registering securities that the Broker intends to offer for exchange with the Market;
- obtaining price quotes for those securities;
- posting current security offers with the Market; and
- maintaining a list of unacknowledged (open) orders from the investors.

The protocols are augmented to facilitate the above activities. A Broker cooperates with its decision-making counterpart, *Broker Decision Support role*, to obtain price quotes for the securities it offers for exchange. Once the quotes are generated, the Broker forwards them to the quotation system. When an order is received, a Broker updates its portfolio accordingly and puts the order on the list of unacknowledged orders. The Broker then notifies its client, the Investor, about the fulfillment of the order. When the receipt of the order is acknowledged by the Investor, the Broker commits the portfolio update by removing the order from the list of unacknowledged orders. Brokers are granted write permission for the offer posting system. Their read/write permissions allow to manipulate internal data structures containing their security offers and orders from Investors.

A broker exercises direct control over security prices. The investors influence the security price evolution through their purchasing activity. The Investor and Broker roles are provided with their decision making counterpart, *Decision Support* (DS), whose responsibility is to supply the desired portfolio data to the transactional counterpart. The Broker DS role also supplies the price quotes for the securities offered for exchange by the Broker. Its activity is the computation providing the above data, and the protocols it supports are: provide the security price (for the Brokers) and the desired volumes for tradable securities. Reasoning capabilities, knowledge and beliefs about the market are under the purview of this role.

Interaction Model The interaction model specifies message interchange in the system. The interactions occur between the Broker and the Market roles, the Investor and the Market, the Broker and its Decision Support, the Investor and its Decision Support, and the Investor and the Broker. The nature of these interactions as well as some operations internal to the roles and triggered by external messages are described below.

As is mentioned above, the Market role maintains a pool of registered securities - the securities available for trading. All the securities but the money market are offered for exchange by the broker community. The Market role is responsible for the money market; this security cannot be offered by a Broker. It performs the function of the *numeraire*, or the riskless security.

For every new security a Broker wants to offer it sends a message to the Market with the request to register the security. When the security is registered a designated area is created within the quotation system to accommodate further price quotes and the Broker is granted a write permission to the this area. A similar message exchange happens when a Broker takes upon itself to offer a registered security that it has not offered before; except that in this case there is no need to create a new quotation area and the Broker receives a write permission to the established area allocated to the security.

Price quotes are generated by the Broker Decision Support roles and conveyed to their respective Broker roles. Once the Broker obtains the price quote, it conveys the quote to the offer posting service or quotation system within the Market.

Investors query the quotation system to obtain price quotes. The quotes are forwarded to their respective Decision Support roles. Once an Investor Decision Support role has generated desired portfolio information, it forwards the data to its transactional counterpart, the Investor role. The Investor role communicates corresponding orders to a number of chosen Brokers. An order is issued for each individual security and sent to the broker with the best deal. The Broker then notifies its client, the Investor, about the fulfillment of the order. The Investor acknowledges the receipt of the security to the Broker.

The procedure described above makes several assumptions about the rights and responsibilities of the agents in the system and the communication infrastructure. The first assumption is that orders are binding; a Broker must fulfill an order that it receives, and the Investor cannot withdraw an order once it has been placed. The second assumption is that no messages are lost or indefinitely delayed by the system. This is a simple model serving as a basis for further elaboration. A Broker might not be able to fulfill all of the orders it receives due to certain portfolio constraints that could be imposed on it by the Market. If this happens, the Broker must remove or update its offer information with the Market, and notify the Investors about the change. The Investors roll back the provisional changes in their portfolios corresponding to returned orders. A system of penalties for the Brokers and restitutions to the Investors need to be implemented to prevent this from happening. These decisions are application-specific and should be made on a case-by-case basis.

A market order should generally remain binding for the Investor; a portfolio constraint check must be made while the provisional changes to the portfolio are carried through, prior to the issue of the order. Note, that the changes made to the Investor's portfolio are determined solely by its corresponding Decision Support while in the Broker's case its Decision Support cannot completely determine what changes to the Broker's portfolio will be requested by the investor community: it faces uncertainty about Broker's portfolio updates and can only make predictions/recommendations.

Another consideration should be given to a case of imperfect communication infrastructure. For instance, this may be the case in a large distributed simulation. In such a simulator certain rules have to be implemented to resolve the issue of outdated requests and timeouts. This issue has been exhaustively covered by the mainstream database research community [HV99]. Once a transaction has been initiated, agents may utilize protocols developed for distributed database transaction processing to achieve a mutually consistent state of their respective portfolios.

4.3 Design

Now we proceed through unification of some abstract entities to achieve concrete ones and further translate them into their implementation counterparts. As will be seen below, we unify transactional and decision-support roles to obtain a reasoning and acting agent, a market participant. When translating this design into a concrete implementation it is desirable to keep the agent architecture highly modular so that one type of decision support can be easily exchanged for another depending on the application and user preferences. Certain applications call for adaptable agents, i.e., the agents that must modify their decision-making procedure to accommodate changes in the environment. These decisions are application-specific and should be treated on a case-by-case basis.

Agent Model Following the idea outlined above, we unify the Investor role with its Decision Support role to obtain an Investor agent, and the Broker role together with its Decision Support role yield a Broker agent. These are the market participants populating the simulated environment. There could be an arbitrary number of these agents per instantiation depending on the user choice and computational resources available. We will further refer to these agents as Investors and Brokers.

The Market agent encompasses only one role, the Market role. This agent is unique for each instantiation of the simulator and performs the function of the centralized administrative and regulatory authority. It regulates the marketplace and provides the means to the Investors and the Brokers to find each other.

Service Model The service model describes the interactions on the agent level and is derived directly from the Interaction Model for the roles.

The Broker agents contact the Market agent to register new securities and to post quotes for tradable securities. The Investor agents direct their price queries to the Market agent to obtain current quotes for registered securities. Once an Investor decides to initiate a transaction it sends an order message to a set of chosen Brokers. Depending on whether the market model holds the orders binding for the brokers, the Broker either fulfills the received order

or notifies his client about his inability to do so. The number and semantics of the messages exchanged in order to complete or abort the transaction between the Investor and the Broker may differ depending on the protocol utilized to achieve a mutually consistent state of their respective portfolios.

Acquaintance Model This model represents communication channels in the simulated market. Observe that, until an Investor is ready to initiate a transaction, there is no communication between the Investor and a Broker. Also, individual Investors do not communicate with each other. All the information available to an Investor is contained in the quotation system. Brokers have access to their respective trade histories. In most applications, Brokers will be assumed competitive and will not share their private trading histories.

Once an Investor wishes to make changes to its portfolio it obtains relevant offer information from the Market. All the message traffic concerning securities exchange happens between the Investor and a chosen Broker. Brokers communicate to the Market to post their offers or register new securities as well as to obtain information from the quotation system. Note that, since a Broker is also an Investor, it may initiate transactions with other Brokers. The graph corresponding to this model is presented in Figure 3.

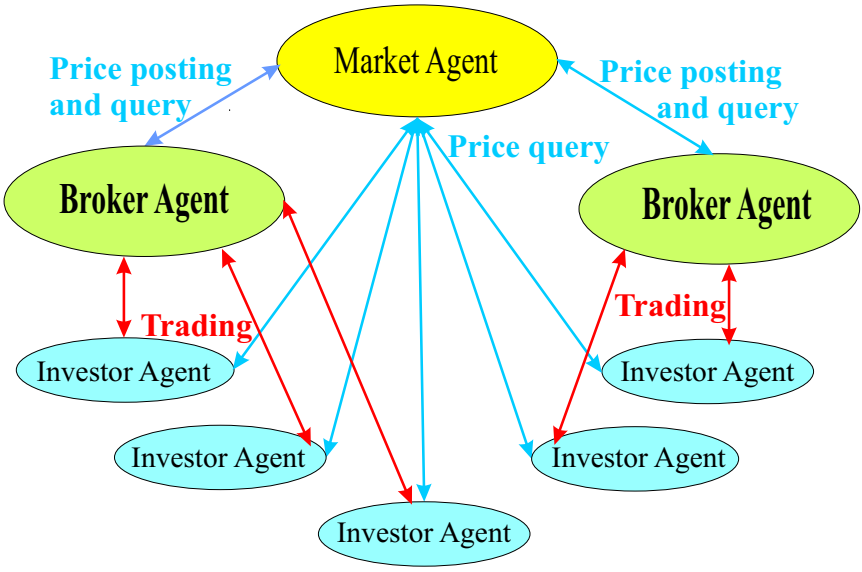


Fig. 3. Acquaintance model graph

User's Perspective The operations of the tranasactional part of the simulator are modeled by the general trading rules of a dealer market, and therefore

can be assumed to apply without major change through a number of different applications. The decision support used by the agents is highly application-specific and may vary even within the same application. For example, stock price evolution in a community of technical traders can be studied under a variety of diverse technical trading rules. In the real world, a market is populated by market participants governed by the same protocols in their transactions, but individual reasoning in their decisions to trade. The strength of the multi-agent approach lies in the ability to populate a simulated market with agents heterogeneous with respect to their decision support, and study the emerging complexity.

In the market model described above, where the market community is stratified into investors and brokers, there are two types of decisions to be made. A security price quote needs to be calculated (this action is available only to the Broker agents), and portfolio rebalancing information needs to be generated (this is done by both Brokers and Investors). These decisions are made based on the following:

- information available from the Market, such as current securities prices and the money market interest rate, as well as, possibly, price and interest rate history;
- the agent's internal beliefs about the market properties.

The above considerartions suggest that it is easy to develop a standard interface between the transactional and the decision-support components of the agent architecture, since the inputs and the outputs are determined. Thus, decision support becomes the only application-specific component of the simulator. While market information is obtained by the Investor role, the agent's beliefs are confined to the Investor/Broker Decision Support role. Therefore, a market agent should provide a standard interface to the market-wide information to be utilized by its decision support.

From a user perspective, a market simulation proceeds through a number of steps. *Agent Creation* step instantiates the model. The Market agent is created and the simulation horizon is set. The marketplace is populated with Broker and Investor agents, which receive only their most basic properties, such as their identification numbers. Further, the simulator requires a set of initial data which is supplied at the Data Generation step:

1. *Portfolio Definition* is concerned with specifying portfolio constraints (e.g., short-selling and borrowing constraints) and assigning an initial portfolio for each Broker or Investor agent. The initial portfolio may reflect a position in the money market as well as other types of securities.
2. *Decision support and learning* endows the agents with their respective reasoning capabilities. Decision support modules are chosen from the set of available options or supplied by the user .

Notice that every agent in the system is initialized individually. This implies that even within the same type of agent, like a Broker or an Investor

agent, a great variety of personal features can be implemented, like distinct utility functions (if optimization is used as decision support) and market constraints, different pricing algorithms, and so on. The simulator aims to recreate a variety of investor types and any number of investors and brokers, as well as to accommodate an array of market models.

Once the setup is over the user starts the simulation. The simulation is governed by the simulation clock, which triggers price updates by the Brokers and portfolio re-evaluation by the trading community. The *Recorder* component performs the necessary data capture and visualization. The price history is made available for the market participants. The transaction log is not seen by the agents. Broker agents may maintain their own transaction histories, which, in most applications, will not be shared among the Brokers (remember the competitiveness assumtion). However, certian studies may target broker coalition formation or other types of cooperative behavior. In this case, the user may modify the permissions on the inidividual transaction histories to accomodate such behavior.

The simulation ends when the simulation clock reaches the horizon. Agent success is measured through the value of the terminal portfolio. Establishing an unambiguous procedure to value the resulting portfolio is a challenge to the decision support designer. If the terminal portfolio consists of securities for which the market community agrees on a price, i.e., highly liquid securities, the portfolio can be valued against the market. If there are thinly traded securities in the agent portfolios the valuation becomes subjective. The latter situation can be avoided by ensuring that such illiquid securities expire before the simulation ends.

4.4 Implementation

The methodology presented above was employed in the development of MAFiMSi (Multi-Agent Finanacial Market Simulator), a general-purpose finacial market simulator of a dealer-type market. MAFiMSi is implemented as a library of C++ classes that currently support a stand-alone market simulation. C++ was chosen for easy compatibility with numerical methods software, which is predominantly written in C, specifically with the IBM Optimization Solutions and Library [IBM04].

The sourse code of the simulator is available at
`http://www.cs.umbc.edu/~streltch/mafimsi.html`.
The simulation is guided by a driver. A simulation driver should supply agent creation and data generation, which is combined into a single step, as well as simulation clock updates which are sent to the agents. A user is expected to supply decision support modules to the investors and brokers depending on the particular microeconomic model employed. A pointer to the corresponding decision support function is supplied to each market participant agent as it is instaciated, or immediately after. Unless this pointer is initialized within the agent, it will not be able to participate in market activities.

A sample simulation driver is supplied for a synchronous simulation. In such a simulation, as time advances to the next trading date, all the Brokers in the market are invited to submit their quotes, and then Investors make their decisions besed on the latest information from the Brokers. All the orders are completed before the simulation is allowed to proceed to the next trading date. The simulation is over once the clock reaches the horizon.

4.5 Usage Example: Simulating a Derivatives Market

Understanding the ties between derivative securities pricing and trading incentives for both parties of the trade is crucial to the development of a mature finance theory. The key to it lies in modleing of the trading parties as heterogeneous entities that are motivated to trade in order to meet their previous obligations or hedge their future exposures. While it is prohibitatively hard to theoretically model heterogeneity of the investor community, an agent-based system offers ample opportunities for experimantal research. Below we describe an experimental environment that can be employed for such a study.

To concentrate only on the aspect of price formation for the derivative securities, the simulation does not reproduce the price formation mechanism for the underlying assets. Instead, following an established procedure of computational finance, it models the evolution of the prices of the underlying via a stochastic process [KS98].

In the real world, the price of an asset changes discretely (by a tick) after a fixed time interval since the last price adjustment (also called a tick). However, one cannot incorporate all possible discrete states into a simulated environment: the model size grows exponentially and the computation becomes intractable. Mathematically, continuous time/continuous space models serve to emulate real world granularity. These models work if a closed-form analytical solution exists. Since, in the general case, such a solution can not be found, an alternative approach is to employ a discrete time/discrete space model which bundles several events into one. The discrete setting is of coarser than real-world granularity, which makes the model amenable to computational techniques. We follow the latter approach and use a discreet stochastic procee (a scenario tree) to model the movements of the underlying.

An example of a scenario tree for a discrete stochastic process with a finite time horizon is given in Figure 4. Each node represents a possible state of the price process for the underlying at a particular time step. Every state except the initial one has a unique parent, and every non-terminal state has a set of child states. The outgoing edges of a particular state in the tree connect it to all possible states of the price process evolution for the next time step, provided that the given state occurs. Each path from the root to a leaf node corresponds to a single scenario. The probability of each scenario is a product of the probabilities on the edges corresponding to it. For further details of the mathematical modeling of market processes we refer the reader to [HP81].

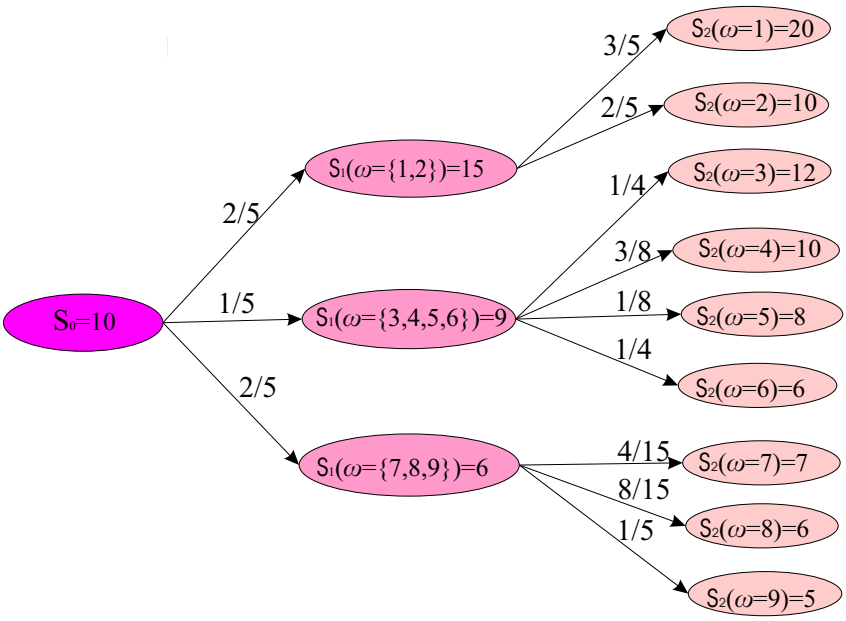


Fig. 4. A scenario tree

To supply the underlying prices to the rest of the agent community we introduce a single Broker agent, called *Stock Broker*, whose Decision Support component performs a random walk through the scenario tree to establish the underlying price vector at each trading date. An additional setup step, namely *Scenario Definition*, becomes a part of the data generation step for the Stock Broker.

The interactions in a synchronous derivatives market simulator proceed as follows. The Stock Broker generates a vector of the basic securities prices and enters the quotes into the quotation system. A Broker trading in derivative securities observes the prices of the underlyings, and decides on the derivatives prices according to its private valuation algorithm. Investors obtain price quotes for both the underlying assets and the contracts written on them (derivatives) and make their portfolio rebalancing decisions. The brokers provide liquidity to the market, i.e., satisfy the investors' requests.

The goal of such experiments is to provide a testbed for various contract pricing techniques and to observe market dynamics depending on the properties (goals) of the market participants.

Further possible applications may include emerging market, such as electricity and bandwidth exchanges, etc. These types of markets are of special interest because of the lack of established decision support techniques and the need to experimentally validate the proposed ones.

5 Conclusions

Given a financial market, there are two main reasons to simulate it. One is to provide a testbed for the components (such as price discovery) necessary to fully automate the market. The other is to provide an experimental setting to observe the consequences of a range of investor behaviors. The second case can be thought of as a special case of the first (since decision support is one of the components required for an automated market), but even if market automation were not a goal, an experimental environment would still be desirable.

Once an order is placed, there are three stages to a financial market transaction: order routing, execution, and clearing and settlement. The first and last of these have been fully automated, while the negotiation and/or decision necessary for execution is automated only in limited situations. In general, decision support, whether for an investor or a market maker, is still to be automated. Research here has primarily focused on the investor employing AI techniques such as genetic algorithms. These have been used to build agents that engage in technical trading with the objective of maximizing their profit from market transactions. Institutional investors, however, do not engage in technical trading, and, for the most part, are more interested in managing risk than in maximizing return on their market investments. Automation of the market maker's decision support has not yet received any attention.

To facilitate experimental study of a wide range of financial market problems, we propose a unified approach to simulator construction which recognizes the objects common to all financial markets. Our approach decomposes the complexity of financial markets into relatively simple tasks and processes, and clearly separates the operational complexity of financial markets from the decision support that drives market agents. Our approach enables the design or simulation of a variety of markets, as well as the deployment of agents with a variety of decision support mechanisms. We ourselves have simulated a derivative securities market using this approach, and are using it to study derivative price formation.

We see two main directions on the research agenda for the area of multi-agent simulation of financial markets. One direction elaborates on the system and agent architecture, functionality, and properties. The goal is twofold: to improve the modeling potential of such systems, and to enable the incorporation of software agents into real-world financial marketplaces (i.e., to automate marketplaces). The other direction concentrates on elucidating the patterns that result from the interaction of heterogeneous market agents. We feel that our proposed reference architecture will be useful in both pursuits.

References

[AHL+97] W. B. Arthur, J. H. Holland, Blake LeBaron, R. G. Palmer, and P. Tayler, *Asset pricing under endogenous expectations in an artificial stock market*, The Economy as an Evolving Complex System II (Menlo Park, CA) (W. B. Arthur, D. Lane, and S. N. Durlauf, eds.), Adisson-Wesley, 1997.

[BDD00] Christopher H. Brooks, Edmund H. Durfee, and Rajarshi Das, *Price wars and niche discovery in an information economy*, Proceedings of ACM Conference on Electronic Commerce (EC-00) (Minneapolis, MN), October 2000.

[FJ97] Sergio Focardi and Caroline Jonas, *Modeling the market: New theories and techniques*, Frank J. Fabozzi Associates, New Hope, Pennsylvania, 1997.

[FSW00] Ming Fan, Jan Stallaert, and Andrew B. Whinston, *The internet and the future of financial markets*, Communications of the ACM **43** (2000), no. 11, 82–88.

[HP81] J. Michael Harrison and Stanley R. Pliska, *Martingales and stochastic integrals in the theory of continuous time trading*, Stochastic Processes and their Applications **11** (1981), 215–260.

[HV99] Tamer M. Özsu and P. Valduriez, *Principles of distributed database systems*, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[IBM04] IBM, *IBM Optimization Solutions and Library*, 2004, http://www-3.ibm.com/software/data/bi/osl/index.html.

[IU99a] Kiyoshi Izumi and Kazuhiro Ueda, *Analysis of dealers' processing financial news based on an artificial market approach*, Journal of Computational Intelligence in Finance **7** (1999), 23–33.

[IU99b] ———, *Analysis of exchange rate scenarios using an artificial market approach*, Proceedings of the International Conference on Artificial Intelligence (A. Amin, C.-H. Chen, and et. al, eds.), CSREA Press, 1999, pp. 360–366.

[JPB99] S. Joshi, J. Parker, and M. A. Bedau, *Technical trading creates prisoner's dilemma: Results from an agent-based model*, Computational Finance (Cambidge, MA) (Y. S. Abu-Mostafa, Blake LeBaron, A. W. Lo, and A. S. Weigend, eds.), The MIT Press, 1999.

[KHG00] Jeffry O. Kephart, James E. Hansen, and Amy R. Greenwald, *Dynamic pricing by software agents*, Computer Networks **32** (2000), no. 6, 731–752.

[KHL+98a] Jeffrey O. Kephart, James E. Hanson, David W. Levine, Benjamin N. Grosof, Jakka Sairamesh, Richard B. Segaland, and Steve R. White, *Dynamics of an information-filtering economy*, Proceedings of the Second International Workshop on Cooperative Information Agents, July 1998.

[KHL+98b] Jeffry O. Kephart, James E. Hansen, D. W. Levine, Benjamin Grosof, J. Sairamesh, R. Segal, and S. R. White, *Emergent behavior in information economies*, Proceedings of the International Conference on Multi-Agent Systems, 1998.

[KS98] Ioanis Karatzas and Steven Shreve, *Methods of mathematical finance*, Springer-Verlag, new York, New York, 1998.

[LAP99] Blake LeBaron, W. B. Arthur, and R. G. Palmer, *The time series properties of an artificial stock market*, Journal of Economic Dynamics and Control **23** (1999), 1487–1516.

[LeB00] Blake LeBaron, *Agent-based computational finanace: Suggested readings and early research*, Journal of Economic Dynamics and Control **24** (2000), 679–702.

[LeB01a] ———, *A builder's guide to agent-based financial markets*, Quantitative Finance **1** (2001), 254–261.

[LeB01b] ———, *Evolution and time horizons in an agent based stock market*, Macroeconomic Dynamics **5** (2001), 254–261.

[MGM99] P. Maes, R. Guttman, and A. Moukas, *Agents that buy and sell*, Communications of tha ACM **42** (1999), no. 3, 81–91.

[PAHL94] R. G. Palmer, W. B. Arthur, J. H. Holland, and Blake LeBaron, *Artificial economic life: a simple model of a stock market*, Physica D **75** (1994), 264–274.

[Par97] Frank Partnoy, *Fiasco the inside story of a wall street trader*, Penguin Books, New York, 1997.

[PBR95] Arnold Picot, Christine Bortenlänger, and Heine Röhrl, *The automation of capital markets*, Journal of Computer-Mediated Commerce **1** (1995), no. 3. Available online at http://www.ascusc.org/jcmc/-vol1/issue3/picot.html.

[RN95] S. Russell and P. Norvig, *Artificial intelligence*, Prentice Hall, New Jersey, 1995.

[Rus96] John Rust, *Dealing with the complexity of economic calculations*, 1996, Workshop on Foundamental Limits to Knowledge in Economics.

[San99] Tuomas Sandholm, *eMediator: A next generation electronic commerce server*, AAAI Workshop Technical Report WS-99-01 (Orlando, Fl), AAAI Workshop on AI in Electronic Commerce, 1999, pp. 46–55.

[SEC99] Securities and Exchange Commission, *Securities and Exchange Commission special study: On-line brokerage: Keeping apace of cyberspace*, November 1999, Available online at http://www.sec.gov/news/studies/cyberspace.htm.

[SNY01] Olga Streltchenko, Nanjangud C. Narendra, and Yelena Yesha, *A reference architecture for multi-agent simulation of derivatives markets*, Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Applications (Bangor, Wales), 2001.

[Tes02] Leigh Tesfatsion, *Notes on the Santa Fe Artificial Stock Market model*, Available online at http://www.econ.iastate.edu/classes/econ308x/tesfatsion/sfistock.htm, 2002.

[TK98] Gerald J. Tesauro and Jeffrey O. Kephar, *Foresight-based pricing algorithms in an economy of software agents*, Proceedings of International Conference on Information and Computation Economies, October 1998.

[Var98] Hal Varian, *Effect of the internet on financial markets*, 1998, Available online at http://www.sims.berkeley.edu/∼hal/Papers/brookings-paper.html.

[WJK00] Michael Wooldridge, Nicholas Jennings, and D. Kinny, *The gaia methodology for agent-oriented analysis and design*, Journal of Autonomous Agents and Multi-Agent Systems **3** (2000), no. 3, 285–312.

[Zar99] Frank G. Zarb, *The coming global digital stock market, speech at the National Press Club*, 1999, Available online at http://www.nasdaqnews.com/views/speech/digmarkets.htm.

[ZJW00] Franco Zambonelly, Nicholas Jennings, and Michael Wooldridge, *Organizational abstractions for the analysis and design of multi-agent systems*, Proceedings of the Firstst International Workshop on Agent-Oriented Software Engineering (Limerick, Ireland), 2000, pp. 127–141.

Adaptive Agents in Coalition Formation Games

Alex K. Chavez

University of Pennsylvania, Philadelphia, PA 19004, USA,
achavez@sas.upenn.edu

Abstract. Coalition formation games form an important subclass of mixed-motive strategic situations, in which players must negotiate competitively to secure contracts. This paper compares the performance of two learning mechanisms, reinforcement learning and counterfactual reasoning, for modeling play in such games. Previous work [CK04] found that while the former type of agent converged to theoretical solutions, they did so much more slowly than human subjects. The present work addresses this issue by allowing agents to update extensively based on counterfactual reasoning.

1 Introduction

Traditional solution concepts for n-person characteristic function games give stable sets of payoff assignments under certain rationality assumptions, but they have little to say about the process by which players arrive at these payoff sets. Furthermore, while it is possible that players realize these stable sets via deliberation, early experimental evidence [KR74] suggested that players engage in a learning process over repeated trials.

An explicit account of such a learning process can help to elucidate these problems. Modeling such a process, however, is complicated by the very large number of actions players must choose from. For example, players decide whether to accept coalition proposals, presumably based on the offer amount (drawn from an interval of the rational or real numbers), the offerer, and the other members of the coalition; they also decide whom to propose coalitions to, and how much to offer each member in the proposed coalition. To address these issues, Dworman, Kimbrough & Laing used genetic programming over the space of 1) the set of players, 2) lower and upper bounds of acceptance values, and 3) offer amounts.[1] They found that when mean payoffs of their system had settled, agents' payoffs approximated the game's quota values (a solution concept to be discussed).

While population models can be useful for exploring issues in agent-based systems and for representing evolutionary games, their interpretation for interactions on a smaller time scale (such as those of human players in repeated

[1] [DKL95a], [DKL95b], [DKL96], and [DKL96]

trials) is limited. Firstly, the timescales are different, and learning tends to be too slow. Secondly, agents in population models cannot be considered to be "minimally rational," as they are usually bare strategies (e.g., in replicator dynamics). Thus, it may be important to model human players as (boundedly) rational agents.

Borrowing from the approach of [ER98] in normal form games, [CK04] applied an individual learning model to five 3-person coalition games of [KR74]. They used a simple linear updating scheme (of reinforcement learning, see Equation 2) to update "aspiration levels" [MF02] of agents. These values, which remain within payoff bounds, represent the payoffs each agent believes it can get from other agents in a given coalition. Aspirations are mapped into behaviors by either a "greedy" rule, under which the agent maximizes over aspirations to the other agents, or a "matching" rule, under which the agent chooses probabilistically in ratios that match aspirations. [CK04] found that agents using the greedy rule converged to quota solutions in all five games, while matching agents never did. However, convergence took much longer for agents compared to human subjects in [KR74], who reached quotas in about four trials. This difference may have been due to calibration by subjects during practice rounds for which [KR74] did not report data, to deductive or counterfactual reasoning (reinforcement learning only uses information about received payoffs), or to experimental manipulations of a communication condition.

This paper compares the performance of a reinforcement learning model to that of one which incorporates information about payoffs *not* received, as well as payoffs received. It also examines the performance of agents utilizing a variety of offer behaviors. The remainder of the paper is organized as follows. Section 2 reviews coalition formation and presents the games studied in this paper. Section 3 discusses the learning model, updating equations, and offer behaviors. Section 4 compares the model's performance against theoretical solutions and again human data. Section 5 concludes.

2 Background and Description

2.1 Coalition Formation

Coalition formation describes a situation in which n players negotiate competitively to secure advantageous contracts. Each player can act alone to attain some fixed payoff. However, by pooling resources with one or more other players and forming a *coalition*, a player can exploit its comparative advantages and thereby negotiate for payoffs that are larger than the sum of the payoffs the individuals in the coalition can get by acting alone.[2] The pay-

[2] This property is known as superadditivity, and holds for all games studied here. The payoff for acting alone is normalized to zero, and the assumption is made that all coalitions have payoffs greater than or equal to zero. Payoffs are assumed to be on the same scale, or in units of "transferable utility" [LR57].

off assigned (exogenously) to a given coalition is known as its *coalition value*, and is the amount guaranteed to that coalition. While it generally is more advantageous for a player to be included in some coalition than to go "solo," it is up each member to secure a portion of the coalition value for herself or himself. Once every member of the coalition agrees to some proposed split, then this agreement becomes binding and is enforced.

While coalitions can have different values, it is not always the case that individuals will gain the greatest payoffs in coalitions with the highest values. Players must consider carefully whom to propose coalitions with, and how much to offer each coalition member. To take an example, let A, B, and C be players in a game with coalition values defined by $v(AB) = 95, v(AC) = 90, v(BC) = 55$, so that if A and B were to form a coalition, they would decide how to split 95 between them. Suppose that A makes an offer first, and proposes a 66/29 split with B (A would get 66). B declines, and offers a split of 30/25 with C. Note that both B and C would do better under B's proposal than under A's proposal. C would do well to accept this proposal, for this reason: *he cannot counteroffer to any other player without that player being able to propose a split with the excluded player that is better for both her and the excluded player*. This special property makes the 30/25 split between B and C *stable* in the sense of [AM64], and the set of such stable allocations is known as the *bargaining set* (see also chapters 3 and 4 of [KR84]).

2.2 Formal Description and Play Procedure

The situation of n-person games with transferable utility described above (a.k.a. coalition formation) falls in the realm of cooperative game theory, and is given by $\Gamma = \langle P, v \rangle$, where P is the set of player labels and v the *characteristic function* mapping non-empty subsets $S \subseteq P$ to the reals (assigning payoffs to each coalition). Specifically, we consider the five games given in Table 1, in which the special property holds that only *pairs* of players have non-zero assignments of payoffs; that is, $v(ABC) = v(A) = v(B) = v(C) = 0$, while $v(AB), v(AC), v(BC)$ are positive (ij denotes the set of players $\{i, j\}$).

The game is played as follows. A player i is selected as the initial offerer, and makes an offer $O_i^j \in [0, v(ij)]$ to another player $j \in P - i$. The offeree j considers the offer, and then either accepts or rejects it. In the case of a rejection, j becomes the offerer for the next round. Instead, if j accepts the offer, then it becomes binding. Remaining players who have not yet committed to a coalition are then allowed to make offers in the same manner. When all players are finished making offers, then the payoffs are disbursed according to the agreements, and the game ends.

If players cannot reach an agreement within a fixed number of rounds, then no coalitions form and all players receive their solo payoffs of zero. Some implementations of the procedure described above include acceptance

Table 1. Characteristic function and quota solutions by game, from [KR74]. $v(ABC) = v(A) = v(B) = v(c) = 0$ for all five games

	Game:				
	I	II	III	IV	V
Characteristic Function					
$v(AB)$	95	115	95	106	118
$v(AC)$	90	90	88	86	84
$v(BC)$	65	85	81	66	50
Quota Values					
ω_A	60	60	51	63	76
ω_B	35	55	44	43	42
ω_C	30	30	37	23	8

and ratification phases, in which tentative agreements only become binding after further opportunities for negotiation.[3]

At the end of the game, each player i receives some fraction x_i of the value of final coalition to which she belongs. The mutually exclusive and exhaustive set of coalitions, $\mathcal{S} = \{S_1, \ldots, S_m\}$, form a partition of P, and is referred to as the game outcome's *coalition structure*. For example, if players A and B form a coalition, and C is excluded, then the coalition structure is $\{AB, C\}$ (which we represent in shorthand as AB, C). Thus, the outcome of the game can be described by the tuple $\langle x; \mathcal{S} \rangle = \langle x_1, \ldots, x_n; S_1, \ldots, S_m \rangle$.

2.3 Quota Games

The particular games considered in the present paper are the 3-person cooperative games shown in Table 1. Because the special property $v(ABC) = v(A) = v(B) = v(C) = 0$ and $v(AB), v(AC), v(BC) > 0$ holds in these games, they are known as *quota games* and have a theoretical solution – the set of *quota values*, which generally are accepted in cooperative game theory (see [KR74,Uhl90] for summaries). The quota value for player i is given by the following equation:

$$\omega_i = .5 \sum \gamma(j,k) * v(jk) \quad \forall j,k \in P, j \neq k \tag{1}$$

where $\gamma(j,k) = 1$ when $i = j$ or $i = k$ and equals -1 otherwise. By way of motivating the quota concept, note that

$$\omega_i + \omega_j = v(ij), \quad i,j \in \{A,B,C\}, i \neq j$$

[3] Our agents engage only in an acceptance phase; we leave ratification for future research.

For example, player A's quota is $\omega_A = .5[v(AB) + v(AC) - v(BC)]$, and $\omega_A + \omega_B = v(AB)$. Table 1 presents quotas values for all five games.

The quota solution is analogous to the set of equilibrium payoffs in non-cooperative games, and is stable for players who follow a weak set of rationality conditions (e.g., of the bargaining set). If the coalition ij is proposed with an agreed split where i's payoff is larger than his quota, j can make a counteroffer to k where both j and k do better. On the other hand, if the proposed coalition is already at the quota solution, then neither player in the coalition can propose a stable split in which he or she is getting a higher payoff. In short, quota splits within coalitions are attractive payoff configurations which represent stability. For this reason, these games elicit much interest in behavioral studies, and there is an extensive experimental literature on humans. The present paper extends the algorithmic literature in these games using various learning models.

3 Learning Models

3.1 Basic Model

The model assumes that players hold beliefs known as *aspiration levels*[4] about the payoffs they can elicit from other players, and that they update these beliefs based on information they receive from playing the game.

The first updating mechanism is that of reinforcement learning,[5] in which players adjust their aspiration levels towards others only when they take an action in the game resulting in a reward. This is in contrast to the second updating mechanism (i.e., for counterfactual reasoning), which assumes that players consider rewards actually received and also rewards *not* received. This occurs when a player has some new evidence of the future reward she can receive based on another player's current action. If a player j rejects an offer from player i, for example, then j knows the *foregone* reward she could have received. Moreover, even if j is not involved in an offer, she may still be able to update her beliefs based on another player i's behavior (the offer amount, in the case that i is the offerer; and the decision to accept or reject, in the case that i is the offeree). Of course, j cannot always update her beliefs. For example, if an offeree k rejects the offer $O_i^k(t)$ from player i, and j's aspiration level to k is less than $O_i^k(t)$, then she has gained no new information and hence does not update. But on the other hand, if her aspiration level to k is *greater than* $O_i^k(t)$, then she can infer from k's rejection that she should lower her aspiration level.

The differences in these updating rules mark two distinct levels of boundedly rational play. In the first, the player is a statistical learning machine

[4] [MF02]

[5] See [KLM96,SB98] for overviews.

who treats the game as a black box offering different rewards (possibly non-stationary) over available actions. While the model might seem overly simplistic for a human player, it has been applied successfully in many normal-form games.[6] In contrast, the second model assumes the player is a sophisticated learner who assigns some level of rationality to the other players of the game. Such players apply deductive and counterfactual reasoning about rewards not received, thus better calibrating their aspiration levels. While play still incrementally moves toward the equilibrium solution, the players in this model are closer to the fully rational ones espoused by static solutions.

3.2 Reinforcement Learning

The first updating mechanism is that of reinforcement learning, in which, after each decision, agents receive a reward from the environment and update their aspiration levels by adding a fraction of the difference between the reward and the previous aspiration level (for the agent with whom it interacted in that round). Specifically, an offering agent updates based on the received reward which equals the portion of the coalition value it secured (in the case of an acceptance), or zero (in the case of a rejection). The receiving agent also updates if it accepts the offer, but not if it rejects it (a rejection is implicit since the receiving agent now makes an offer as the new offerer). Any agent which did not make or receive an offer (there is only one such agent in the games studied here) does not update its aspiration levels for that time step.

Formally, let $r_i^j(t)$ denote player i's reward from player j at round t, which equals $v(ij) - O_i^j(t)$ if j accepted i's offer that round, and zero otherwise. Then, i's aspiration is updated by:

$$A_i^j(t) = A_i^j(t-1) + \alpha[r_i^j(t) - A_i^j(t-1)] \tag{2}$$

where $\alpha \in (0, 1]$ is the weight given to more recent rewards.[7] This can be seen by expanding Expression 2:

$$\begin{aligned}
A_i^j(t) &= A_i^j(t-1) + \alpha[r_i^j(t) - A_i^j(t-1)] \\
&= (1-\alpha)A_i^j(t-1) + \alpha r_i^j(t) \\
&= (1-\alpha)[(1-\alpha)A_i^j(t-2) + \alpha r_i^j(t-1)] + \alpha r_i^j(t) \\
&= (1-\alpha)^2 A_i^j(t-2) + (1-\alpha)\alpha r_i^j(t-1) + \alpha r_i^j(t) \\
&= (1-\alpha)^t A_i^j(0) + (1-\alpha)^{t-1}\alpha r_i^j(1) + \ldots \\
&\quad +(1-\alpha)\alpha r_i^j(t-1) + \alpha r_i^j(t) \\
&= (1-\alpha)^t A_i^j(0) + \sum_{T=1}^{t} (1-\alpha)^{t-T}\alpha r_i^j(T)
\end{aligned}$$

[6] E.g., [RE95], [ER98], [SV99], and [SV01].
[7] [SB98]

where i's aspiration level to j at time t is a weighted average of the prior $A_i^j(0)$ and the received rewards (the weights are exponentially decreasing as $t \to 0$). In [CK04], we set the value of α to 0.2 (a typical value in the reinforcement learning literature), while noting that convergence in mean coalition payoffs was robust over many values of this parameter.

3.3 Counterfactual Reasoning

Reinforcement learning has proven useful in a large number of contexts outside of game theory, but its primary limitation when applied to strategic contexts is that it ignores knowledge that players have about the game structure. Specifically, it fails to model information such as foregone payoffs, of which players often know the exact value. In the case of coalition formation, it also does not capture a player's cognizance of others' behavior when he or she is not included in the active offer (as explained above). The second model presented here is adapted from [CH99]'s experience-weighted attraction (EWA) model, and mathematically is a special case.[8] [CH99] showed that reinforcement learning and belief learning (e.g., fictitious play, which treats foregone payoffs like received ones) are related closely and can be generalized in their hybrid model. In the present application, when player i makes an offer at time t to player j and gets reward $r_i^j(t)$, i updates by:

$$A_i^j(t) = \frac{\phi * N_i(t-1) * A_i^j(t-1) + r_i^j(t)}{N_i(t)} \tag{3}$$

$$N_i(t) = \phi * N_i(t-1) + 1 \tag{4}$$

where the reward $r_i^j(t)$ equals $v(ij) - O_i^j(t)$ in the case that j accepts the offer, and is zero otherwise. The parameter ϕ is the relative weight placed on more recent observations, and corresponds to the recency parameter α in Equation 2. Along with the prior $N_i(0)$, ϕ determines the "experience weight" for player i, $N_i(t)$.[9] The model thus weights previous aspirations by $\phi * N_i(t-1)/N_i(t)$ and received rewards by $1/N_i(t)$.

If player j accepts the offer from i, then j uses Equations 3 and 4 to update aspirations (with i and j permuted in Equation 3, $r_j^i(t) = O_i^j(t)$, and i replaced by j in Equation 4). If j rejects the offer, however, j updates *by weighting the foregone reward by* $\delta/N_j(t)$:

[8] The EWA model allows for attractions to be cumulated or averaged. This paper sets their parameters $\rho = \phi$, so that attractions are always averaged (as in Equation 2) and thus remain within payoff bounds. This allows for the interpretation of attractions as aspiration levels.

[9] The present model sets the initial experience weight to $N_i(0) = 1$, so that $N_i(t)$ is just a discounted sum of t time steps. That is, $N_i(t) = 1 + \phi + \ldots + \phi^{t-2} + \phi^{t-1} + \phi^t * N_i(0) = \sum_{T=0}^{t} \phi^T$ when $N_i(0) = 1$. Equation 2 is a special case of the EWA model when $\delta = 0$ and $N_i(0) = 1/(1-\phi)$ [CH99].

$$A_j^i(t) = \frac{\phi * N_j(t-1) * A_j^i(t-1) + \delta * r_j^i(t)}{N_j(t)}, \tag{5}$$

where $r_j^i(t) = O_i^j(t)$. (Player j updates her or his experience weight as well; from here on, every aspiration level update is assumed to be accompanied by an experience-weight update for that player).

As mentioned earlier, the left-out player k *also* updates under the counterfactual reasoning model. Upon observing i's offer $O_i^j(t)$ to player j, k calculates the amount i is claiming, $r_i^j(t) = v(ij) - O_i^j(t)$, and imagines that i would offer similarly to k. Thus, k estimates i's counterfactual offer to k as $\hat{r}_k^i(t) = v(ik) - r_i^j(t)$, and updates based on this imagined reward:

$$A_k^i(t) = \frac{\phi * N_k(t-1) * A_k^i(t-1) + \delta * \hat{r}_k^i(t)}{N_k(t)}. \tag{6}$$

Additionally, if j accepts i's offer $O_i^j(t)$, and k's aspiration level to j is *below* $O_i^j(t)$, then k updates by reasoning that he or she would be able to get at least $O_i^j(t)$ from j. Thus, k's estimate of the counterfactual reward from j becomes $\hat{r}_k^j(t) = O_i^j(t)$, and k updates by:

$$A_k^j(t) = \frac{\phi * N_k(t-1) * A_k^j(t-1) + \delta * \hat{r}_k^j(t)}{N_k(t)}. \tag{7}$$

Finally, if j rejects i's offer and k's aspiration level to j is *above* the offer value, then k infers that it is too high. In this case, k updates according to Equation 7 with $\hat{r}_k^j(t) = 0$ (k "imagines" j rejected k's offer). The other two cases do not provide k with any new information (i.e., if k's aspiration level is above $O_i^j(t)$ and j accepts, or if j rejects but $A_k^i(t) \leq O_i^j(t)$), and so k does not update.

The weight δ placed on these counterfactual payoffs can be interpreted as a combination of the player's vividness of imagination and the importance he or she places on counterfactual payoffs [CH98]. When δ is low or zero, then the model is a form of reinforcement learning, where players place little or no weight on counterfactual payoffs; when it is high or one, it is an adaptation of the weighted fictitious play model (see [FL98]), where not only actions for foregone payoffs are reinforced, but also those entirely imagined as in the case of player k above.

3.4 Selection Rules

Aspiration levels represent players' beliefs about the amount of payoff they can elicit from other players. Players can use these values to select whom to offer to in a number of a ways. For example, an offerer can select a player to offer to by maximizing over his aspiration levels. This reflects the traditional

Table 2. Selection rules

Rule	Agent i's probability of selecting agent j
Random	$\frac{1}{\lvert P \rvert}$
Greedy	1 if $A_i^j(t) = \max_{p \in P} A_i^p(t)$ 0 else
ϵ-greedy	$1-\epsilon$ if $A_i^j(t) = \max_{p \in P} A_i^p(t)$ ϵ else
Matching	$\frac{A_i^j(t)}{\sum_{p \in P} A_i^p(t)}$
Softmax	$\frac{e^{A_i^j(t)/\tau^*}}{\sum_{p \in P} e^{A_i^p(t)/\tau^*}}$

* Where P is the set of agents excluding i, and $\tau = \Theta^t$ where t represents the current episode. The parameters ϵ and Θ were set at .01 and .9999 respectively, which are typical and resemble those used in other related studies [SC95]. The positive parameter τ of the Softmax action selection method decreases over time, effecting exploration early on, but becoming greedy as $\tau \to 0$.

economic precept of choice as maximization over beliefs [SV01]. Instead, players might choose whom to offer to in non-maximizing ways which allow for trembles, probability matching [Gal90], or deliberate exploration. The various selection rules examined in this paper which map aspirations levels into actions are given in Table 2.

The Random rule represents a null or "constrained zero-intelligence" [GS93] model. The Greedy rule represents the traditional economic model discussed above, ϵ-greedy its counterpart with trembles. The Matching rule represents probabilistic selection observed in choice experiments with humans and other animals [Gal90], while the Softmax rules balance a mix of searching for better alternatives early on and exploiting later, converging to the Greedy rule as $t \to \infty$ [SB98].

3.5 Offer/Acceptance Behavior

Given the selection rules in Table 2, the natural choice for agent offer and acceptance behavior is for agents to make offers at their aspiration levels, and to accept offers which are greater than or equal to their aspiration levels. While agents could "satisfice" by accepting offers that are below their current aspiration level, such considerations are left to future work.

4 Results

4.1 Fit to Theoretical Values

The updating mechanisms of reinforcement learning and counterfactual reasoning are tested for their explanatory and predictive powers by fitting them to the games in Table 1 in three different ways:[10] 1) individually, 2) over all five games simultaneously, and 3) in a cross-validation scheme. This is done for each combination of updating mechanism and selection rule. Specifically, the mean squared deviation (MSD) of the *mean reward as a winning coalition member* (MRAC)[11] from the respective quota value for each of the three agents is minimized over the appropriate parameter space (α, or δ and ϕ), first for each individual game, then for the best fit over all five games, and finally for each subset of four out of the five games. The last set of chosen parameter(s) is then cross-validated against the fifth game (by predicting the out-of-sample MRAC in the left-out game). This provides two tests of descriptive accuracy and a strict one of prediction. The initial aspiration levels of each agent i to agent j is set to half of their coalition value. MSD values for 300 episodes are shown in Tables 3 and 4 (the corresponding parameter estimates are presented in the Appendix in Tables 7 and 8).

The Softmax rule performs the best, scoring only .26 MSD units over games for the counterfactual reasoning (CR) model in cross-validation (the strictest of the three fit measures). The same rule performs substantially worse, but still well, for the reinforcement learning (RL) model. While they fit well in-sample, both the RL and CR models overfit for the Greedy and ϵ-greedy rules; that is, their out-of-sample performance is poor, as can be seen by their large MSD cross-validation scores. The Random and Matching rules perform poorly both in- and out-of-sample for both models.

By 1,000 episodes, however, both the RL and CR models converge to very small neighborhoods of the quota values (not shown) for any of the greedier selection rules (Greedy, ϵ-greedy, and Softmax), while they never do for the Random or Matching rules. The true test of the two models, then, is to compare their performances on a much shorter timescale of $10 - 20$ episodes – the timescale in which human subjects are able to reach quota values.

4.2 Experimental Data

In their study, [KR74] used human subjects in a computerized experiment designed to examine coalition formation behavior in the games in Table 1. Forty-eight undergraduate male subjects were divided into groups of sixteen

[10] Following [ER98].

[11] Because the quota solution concept is meaningful only for formed coalitions, the quantity traditionally considered [KR74] as a performance measure is the MRAC for each player.

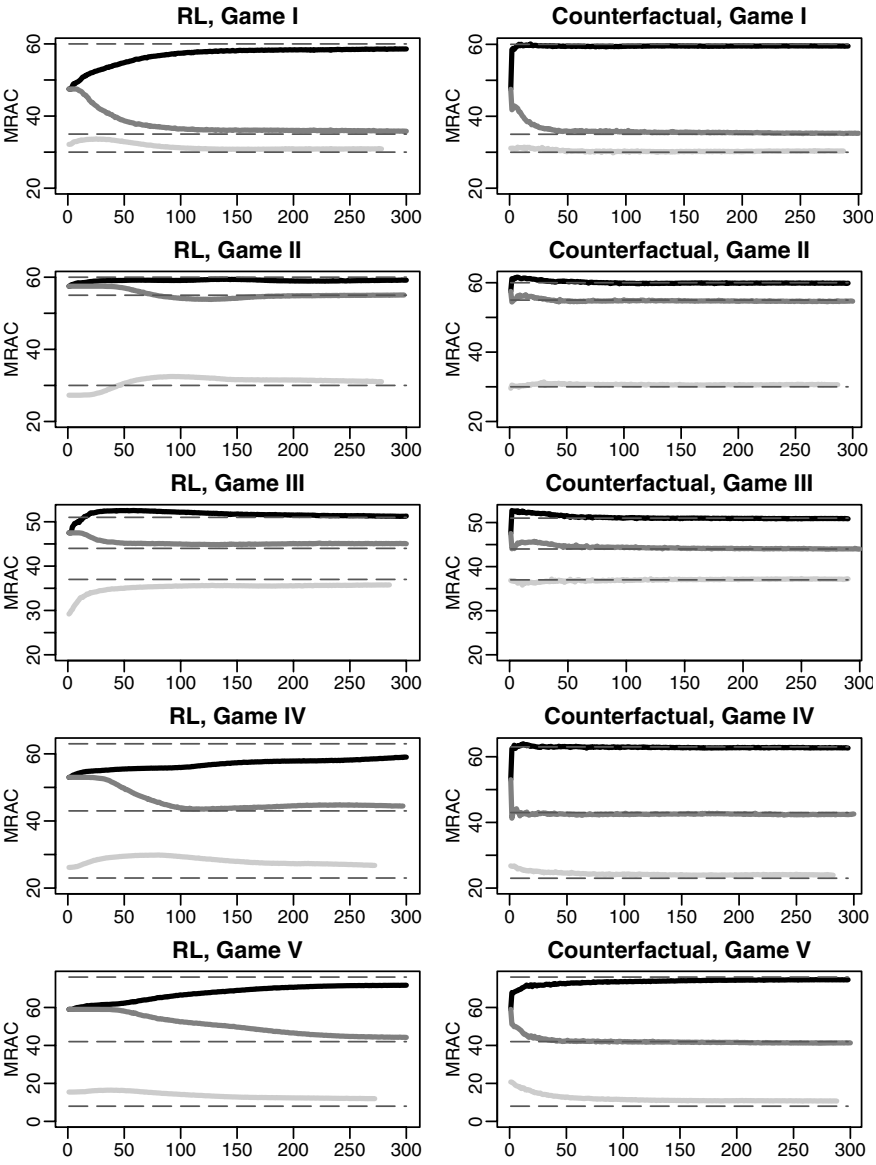


Fig. 1. Convergence over Time of Mean Reward as Members of Winning Coalition (MRAC) for Greedy Reinforcement Learning (RL) Agents and Greedy Counterfactual Reasoning Agents. Black = Agent A's MRAC; Gray = Agent B's MRAC; Light Gray = Agent C's MRAC; Dotted Lines = Players' Respective Quotas

Table 3. MSD between MRAC and quota values for all selection rules and over all games: reinforcement learning model

		Game: I	II	III	IV	V	Over Games
Reinforcement Learning							
Random							
	Individual Fits:	39.98	121.05	21.64	0.98	15.22	119.96
	Cross-Validation:	89.76	138.66	23.14	0.98	75.32	65.57
	Over All Games :	89.76	138.66	23.14	0.98	75.32	65.57
Greedy							
	Individual Fits:	0.14	0.03	0.04	0.11	14.63	6.25
	Cross-Validation:	3.71	0.28	0.44	1.18	15.69	3.66
	Over All Games:	0.17	0.28	0.44	1.18	15.69	3.56
ϵ-greedy							
	Individual Fits:	0.01	0.03	0.05	0.02	13.92	6.33
	Cross-Validation:	2.56	2.46	9.99	3.61	17.18	4.57
	Over All Games:	2.12	0.04	0.99	0.65	17.18	4.19
Matching							
	Individual Fits:	22.55	17.51	3.45	34.99	98.23	43.92
	Cross-Validation:	37.75	26.49	6.92	40.4	125.31	41.83
	Over All Games:	34.93	19.33	6.92	40.4	104.54	41.22
Softmax							
	Individual Fits:	0.35	0.03	0.02	0.78	13.29	6.13
	Cross-Validation:	1.1	0.48	0.4	4.89	18.95	3.77
	Over All Games:	1.1	0.48	0.4	2.28	13.29	3.51

MSD values are for 300 episodes (50 simulations were run for every parameter setting). "Individual Fits" – computed by minimizing the MSD over each game. "Cross-Validation" – computed by minimizing the MSD over all games *except* the one in that column, and then using that value of α to predict the MRAC for the left-out game. "Over All Games" – computed by minimizing the MSD over all games, and then using that value of α to compute the MSD for the game in that column.

Table 4. MSD between MRAC and quota values for all selection rules and over all games: counterfactual reasoning model

		Game: I	II	III	IV	V	Over Games
Counterfactual Reasoning							
Random							
	Individual Fits:	0.11	0	0.06	0	0.21	11.63
	Cross-Validation:	49.9	0.81	14.86	3.44	12.86	5.34
	Over All Games:	1.08	0.81	2.65	3.44	8.55	3.3
Greedy							
	Individual Fits:	0	0	0.06	0.01	0.11	12.92
	Cross-Validation:	0.26	0.3	13.66	49.82	2.36	5.54
	Over All Games:	0.26	0.3	11.06	6.64	2.36	4.12
ϵ-Greedy							
	Individual Fits:	0.02	0.01	0.09	0.03	0.68	0.166
	Cross-Validation:	18.95	1.06	19.69	1.38	20.85	12.386
	Over All Games:	11.08	0.12	2.23	0.66	8.52	4.522
Matching							
	Individual Fits:	1.09	1.15	0.05	7.72	17.79	48.04
	Cross-Validation:	34.39	20.79	4.6	21.4	87.01	16.53
	Over All Games:	1.9	3.76	4.6	21.4	36.23	13.58
Softmax							
	Individual Fits:	0	0	0	0.01	0.8	1.24
	Cross-Validation:	0.05	0.12	0.24	0.01	1.32	0.26
	Over All Games:	0.05	0.12	0.24	0.01	0.8	0.24

MSD values are for 300 episodes (50 simulations were run for every set of parameters). "Individual Fits" – computed by minimizing the MSD over each game. "Cross-Validation" – computed by minimizing the MSD over all games *except* the one in that column, and then using those parameters estimates to predict the MRAC for the left-out game. "Over All Games" – computed by minimizing the MSD over all games, and then using those parameters estimates to compute the MSD for the game in that column.

and participated in three separate experiments. In the first experiment, messages were public, so that all players were aware of the others' offers. Subjects had to send messages publicly in a fixed order (as opposed to being able to speak at will). In the second experiment, messages could be private, but again were sent in order. In the last experiment, messages could be private, but were sent at will. The sixteen players in each experiment were broken up into four quartets, who played each of the five games for four iterations. For each game, one member of the quartet would sit out as an observer – this procedure was employed to allow subjects to reflect upon the task, and to increase the validity of the assumption of independence between games. Order of play between and within games was randomized subject to the condition that no player would be observer in two consecutive rounds. They found that the mean deviation from the quota decreased both within iterations (measured over the first offer made to the winning coalition, the first accepted offer by that coalition, and the ultimately ratified offer by that coalition), and also across iterations, for first offers made to the winning coalitions. Furthermore, they found that the mean deviation from the quota values was not significantly different from zero ($\alpha = .05$).

Despite the fact that subjects were allowed to practice the game prior to playing for monetary rewards and also to communicate (which may have sped up convergence), the small number of rounds in which they reached the quota values is quite remarkable. We noted in [CK04] that the reinforcement learning agents learn too slowly in comparison to the human subjects. To give a sense of comparison of the rate of convergence for the RL and CR agents, Figure 1 shows their MRAC plots over time.[12] Visually, the CR model converges very quickly to the quota values (in less than 50 episodes in many cases), begging the question of whether its performance can match those of humans. The RL agents, on the other hand, take far longer ($200 - 400$ episodes) than humans do.

4.3 Re-estimation on a Short Timescale

The RL and CR models are now re-estimated for the greedier rules (Greedy, ϵ-greedy, and Softmax) for ten episodes. As noted earlier, the original model set each agent's initial aspiration levels to half of its coalition value for each other agent.[13] However, there is reason to suspect that human subjects began learning prior to the recorded experiment during unpaid practice rounds. [KR74] report that the average deviation from the quota of the *first offers*

[12] The parameter estimates from Tables 3 and 4 were used. All other selection rules also show the trend of faster convergence for the CR model compared to the RL one.

[13] This choice was made after finding similar MRAC and MSD values when the initial aspirations $A_i^j(0)$ were drawn from the uniform distribution over $[0, v(ij)]$, which only resulted in more noise.

to the winning coalition is only −2.95 (see Table 5 for the entire list of the deviations over games and coalitions). These small initial deviations reflect pre-game calibration on the part of subjects, possibly due to deliberation or practice. In any case, the model's initial conditions are aligned to those of the humans by setting their initial aspiration levels according to Table 5.[14]

Table 5. Mean deviations from quota of first offers to the winning coalition over games and coalitions (source: [KR74])

	Game: I	II	III	IV	V	Over Games
AB	-5.48	2.16	-0.12	-6.17	-8.62	-4.59
AC	-4.19	4.38	5.56	-4.00	-11.50	-0.74
BC	-1.00	-0.67	1.87	-2.00	NA	0.02
Over Coalitions	-4.51	1.65	2.40	-5.33	-8.98	-2.95

The results for ten episodes for the Greedy, ϵ-greedy, and Softmax rules are shown in Table 6 (1,000 simulations were run for every set of parameters). The CR model again fits the data better overall, due to its extensive counterfactual updating. That is, agents in the CR model update their aspirations levels for foregone payoffs and for offers which they neither made nor received, giving them more chances to move toward theoretical values in every complete round of play (i.e. every time a winning coalition is formed). The RL agents only update once every time they make or *accept* an offer, on the other hand, and hence are far too slow for the rapid convergence to quota values observed in human subjects. The parameter estimates for the CR model, which are shown in the Appendix in Tables 9 and 10, highlight the importance of counterfactual updating. The imagination parameter δ ranges from .54 to 1, with most values above .85. This parameter weights counterfactual payoffs, suggesting that the faster convergence rate of the CR model is due to counterfactual reasoning for these imagined payoffs.

5 Conclusion

Computational modeling is an important tool for investigating the dynamics of negotiation and bargaining in coalition formation. Previous work found strong tendencies toward quota solutions under a simple reinforcement learning framework, but that agents converged too slowly when compared to hu-

[14] For coalition BC in game V, no data is available so the previous initial value of $.5 * v(BC)$ is used.

Table 6. MSD between MRAC and quota values for RL and CR agents: ten episodes

		Game: I	II	III	IV	V	Over Games
Reinforcement Learning							
Greedy							
	Individual Fits:	1.52	2.9	54.64	3.85	0.58	17.26
	Cross-Validation:	5.06	4.97	58.24	7.27	9.63	15.7
	Over All Games:	5.06	3.51	54.64	5.04	9.63	15.57
ϵ-Greedy							
	Individual Fits:	0.88	3.29	55.61	3.87	1.16	17.1
	Cross-Validation:	2.89	3.88	55.61	4.79	8.5	15.13
	Over All Games:	2.89	3.88	55.61	4.79	8.5	15.13
Softmax							
	Individual Fits:	0.74	3.6	15.83	3.65	0.73	12.13
	Cross-Validation:	4.12	5.29	67.15	3.65	9.56	9.75
	Over All Games:	4.12	5.29	15.92	3.65	9.56	7.71
Counterfactual Reasoning							
Greedy							
	Individual Fits:	0	0.01	0.08	0.01	3.4	11.12
	Cross-Validation:	0.39	0.62	7.03	1.93	9.2	2.94
	Over All Games:	0.39	0.44	3.4	1.93	8.2	2.87
ϵ-Greedy							
	Individual Fits:	0.01	0.01	0.1	0.02	3.59	10.73
	Cross-Validation:	2.13	0.5	5.13	3.11	24.8	3.35
	Over All Games:	0.34	0.5	3.32	2.31	6.45	2.58
Softmax							
	Individual Fits:	0	0	0.14	0.02	2.93	12.64
	Cross-Validation:	0.46	0.19	11.32	1.64	15.51	3.2
	Over All Games:	0.46	0.19	3.38	0.88	8.14	2.61
Human:		7.35	6.26	4.41	6.53	45.49	14.01

mans [CK04]. The present work presents a counterfactual reasoning model adapted from [CH99], in which agents update both for received rewards and counterfactual ones. The CR model is able to fit quota values closely for the very short timescale of ten episodes, both in-sample and out-of-sample (where parameters in the latter case are estimated on four out of the five games, and then used to predict the MRAC of the left-out game). The weight placed on counterfactual payoffs in the estimation is high (δ above .85 for most games); this suggests that counterfactual reasoning does play a role in the adaptive process of players.

While the CR model fits well to quota values, another important test is how well it can model human players. Future work might include modeling the full trajectories of individual play, should a detailed data set become available.

In conclusion, the present work extends a simple reinforcement learning model applied to coalition games to include counterfactual reasoning, and shows that agents in this model are able to converge to quota solutions within the very short time frame of ten rounds. Additionally, it demonstrates that systems of artificial agents using relatively simple (and boundedly rational) learning rules can serve as benchmarks for understanding solutions of games. Because these agents are so simple and computationally tractable, this lends additional credence to the quota as a solution concept, and also shows promise for the use of such agents in investigating other characteristic function games.

6 Acknowledgements

The work reported here was supported in part by the University Scholars Fund at the University of Pennsylvania and NSF grant number SES-9709548. The author thanks the Santa Fe Institute for cluster time.

A Parameter Estimates

The tables of this appendix summarize the parameter settings for the experiments described above.

Table 7. Parameter estimates for the reinforcement learning model: 300 episodes

	Game: I	II	III	IV	V
Random:					
Individual Fits:	($\alpha = 0.19$)	($\alpha = 0.02$)	($\alpha = 0.02$)	($\alpha = 0.05$)	($\alpha = 0.11$)
Cross-Validation:	($\alpha = 0.05$)	($\alpha = 0.05$)	($\alpha = 0.05$)	($\alpha = 0.05$)	($\alpha = 0.05$)
Over All Games:	($\alpha = .05$)	($\alpha = .05$)	($\alpha = .05$)	($\alpha = .05$)	($\alpha = .05$)
Greedy:					
Individual Fits:	($\alpha = 0.46$)	($\alpha = 0.72$)	($\alpha = 0.6$)	($\alpha = 0.89$)	($\alpha = 0.82$)
Cross-Validation:	($\alpha = 0.74$)	($\alpha = 0.87$)	($\alpha = 0.74$)	($\alpha = 0.09$)	($\alpha = 0.6$)
Over All Games:	($\alpha = 0.81$)	($\alpha = 0.69$)	($\alpha = 0.69$)	($\alpha = 0.69$)	($\alpha = 0.69$)
ϵ-greedy:					
Individual Fits:	($\alpha = 0.13$)	($\alpha = 0.94$)	($\alpha = 0.14$)	($\alpha = 0.76$)	($\alpha = 0.65$)
Cross-Validation:	($\alpha = 0.65$)	($\alpha = 0.66$)	($\alpha = 0.86$)	($\alpha = 0.65$)	($\alpha = 0.72$)
Over All Games:	($\alpha = 0.13$)	($\alpha = 0.94$)	($\alpha = 0.14$)	($\alpha = 0.76$)	($\alpha = 0.65$)
Matching:					
Individual Fits:	($\alpha = 0.74$)	($\alpha = 0.82$)	($\alpha = 0.85$)	($\alpha = 0.78$)	($\alpha = 0.97$)
Cross-Validation:	($\alpha = 0.97$)	($\alpha = 0.97$)	($\alpha = 0.88$)	($\alpha = 0.88$)	($\alpha = 0.82$)
Over All Games:	($\alpha = .88$)	($\alpha = .88$)	($\alpha = .88$)	($\alpha = .88$)	($\alpha = .88$)
Softmax:					
Individual Fits:	($\alpha = 0.13$)	($\alpha = 0.07$)	($\alpha = 0.26$)	($\alpha = 0.23$)	($\alpha = 0.15$)
Cross-Validation:	($\alpha = 0.15$)	($\alpha = 0.15$)	($\alpha = 0.15$)	($\alpha = 0.1$)	($\alpha = 0.14$)
Over All Games:	($\alpha = .15$)	($\alpha = .15$)	($\alpha = .15$)	($\alpha = .15$)	($\alpha = .15$)

Table 8. Parameter estimates for the counterfactual reasoning model: 300 episodes

	Game: I	II	III	IV	V
Random:					
Individual Fits:	$(\delta = 0.86, \phi = 0.04)$	$(\delta = 0.98, \phi = 0.3)$	$(\delta = 0.96, \phi = 0.08)$	$(\delta = 0.63, \phi = 0.05)$	$(\delta = 0.55, \phi = 0.9)$
Cross-Validation:	$(\delta = 0.92, \phi = 0.01)$	$(\delta = 0.83, \phi = 0.01)$	$(\delta = 0.93, \phi = 0.01)$	$(\delta = 0.92, \phi = 0.01)$	$(\delta = 0.93, \phi = 0.01)$
Over All Games:	$(\delta = 0.92, \phi = 0.92)$	$(\delta = 0.92, \phi = 0.92)$	$(\delta = 0.92, \phi = 0.92)$	$(\delta = 0.92, \phi = 0.92)$	$(\delta = 0.92, \phi = 0.92)$
Greedy:					
Individual Fits:	$(\delta = 0.98, \phi = 0.51)$	$(\delta = 0.97, \phi = 0.4)$	$(\delta = 0.84, \phi = 0.03)$	$(\delta = 0.76, \phi = 0.16)$	$(\delta = 0.99, \phi = 0.28)$
Cross-Validation:	$(\delta = 0.96, \phi = 0.01)$	$(\delta = 0.96, \phi = 0.01)$	$(\delta = 0.97, \phi = 0.01)$	$(\delta = 0.98, \phi = 0.01)$	$(\delta = 0.96, \phi = 0.01)$
Over All Games:	$(\delta = 0.96, \phi = 0.38)$	$(\delta = 0.96, \phi = 0.38)$	$(\delta = 0.96, \phi = 0.38)$	$(\delta = 0.96, \phi = 0.38)$	$(\delta = 0.96, \phi = 0.38)$
ϵ-greedy:					
Individual Fits:	$(\delta = .85, \phi = .04)$	$(\delta = .8, \phi = .9)$	$(\delta = .95, \phi = .08)$	$(\delta = .97, \phi = .18)$	$(\delta = .99, \phi = .26)$
Cross-Validation:	$(\delta = .96, \phi = .01)$	$(\delta = .95, \phi = .01)$	$(\delta = .9, \phi = .01)$	$(\delta = .95, \phi = .01)$	$(\delta = .84, \phi = .01)$
Over All Games:	$(\delta = .95, \phi = .07)$	$(\delta = .95, \phi = .07)$	$(\delta = .95, \phi = .07)$	$(\delta = .95, \phi = .07)$	$(\delta = .95, \phi = .07)$
Matching:					
Individual Fits:	$(\delta = 0, \phi = 0.78)$	$(\delta = 0.01, \phi = 0.36)$	$(\delta = 0.01, \phi = 0.3)$	$(\delta = 0.01, \phi = 0.55)$	$(\delta = 0, \phi = 0.03)$
Cross-Validation:	$(\delta = 0, \phi = 0.01)$	$(\delta = 0, \phi = 0.01)$	$(\delta = 0, \phi = 0.01)$	$(\delta = 0, \phi = 0.01)$	$(\delta = 0.01, \phi = 0.01)$
Over All Games:	$(\delta = 0, \phi = 0.74)$	$(\delta = 0, \phi = 0.74)$	$(\delta = 0, \phi = 0.74)$	$(\delta = 0, \phi = 0.74)$	$(\delta = 0, \phi = 0.74)$
Softmax:					
Individual Fits:	$(\delta = 0.97, \phi = 0.46)$	$(\delta = 0.84, \phi = 0.87)$	$(\delta = 0.96, \phi = 0.63)$	$(\delta = 0.97, \phi = 0.15)$	$(\delta = 0.99, \phi = 0.31)$
Cross-Validation:	$(\delta = 0.99, \phi = 0.01)$	$(\delta = 0.99, \phi = 0.01)$	$(\delta = 0.99, \phi = 0.01)$	$(\delta = 0.99, \phi = 0.01)$	$(\delta = 0.98, \phi = 0.01)$
Over All Games:	$(\delta = 0.99, \phi = 0.31)$	$(\delta = 0.99, \phi = 0.31)$	$(\delta = 0.99, \phi = 0.31)$	$(\delta = 0.99, \phi = 0.31)$	$(\delta = 0.99, \phi = 0.31)$

Table 9. Parameter estimates for the reinforcement learning model: ten episodes

	Game:				
	I	II	III	IV	V
Greedy					
Individual Fits:	($\alpha = 0.1$)	($\alpha = 0.58$)	($\alpha = 0.72$)	($\alpha = 0.75$)	($\alpha = 0.11$)
Cross-Validation:	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)
Over All Games:	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)	($\alpha = 0.72$)
ϵ-Greedy					
Individual Fits:	($\alpha = 0.11$)	($\alpha = 0.68$)	($\alpha = 0.67$)	($\alpha = 0.76$)	($\alpha = 0.12$)
Cross-Validation:	($\alpha = 0.67$)	($\alpha = 0.62$)	($\alpha = 0.62$)	($\alpha = 0.51$)	($\alpha = 0.67$)
Over All Games:	($\alpha = 0.67$)	($\alpha = 0.67$)	($\alpha = 0.67$)	($\alpha = 0.67$)	($\alpha = 0.67$)
Softmax					
Individual Fits:	($\alpha = 0.11$)	($\alpha = 0.65$)	($\alpha = 0.59$)	($\alpha = 0.75$)	($\alpha = 0.11$)
Cross-Validation:	($\alpha = 0.75$)	($\alpha = 0.75$)	($\alpha = 0.11$)	($\alpha = 0.75$)	($\alpha = 0.75$)
Over All Games:	($\alpha = 0.75$)	($\alpha = 0.75$)	($\alpha = 0.75$)	($\alpha = 0.75$)	($\alpha = 0.75$)

Table 10. Parameter estimates for the counterfactual reasoning model: ten episodes

	Game: I	II	III	IV	V
Greedy					
Individual Fits:	$(\delta = 0.94, \phi = 0.32)$	$(\delta = 0.93, \phi = 0.59)$	$(\delta = 0.71, \phi = 0.24)$	$(\delta = 0.54, \phi = 0.81)$	$(\delta = 0.96, \phi = 0.16)$
Cross-Validation:	$(\delta = 0.91, \phi = 0.01)$	$(\delta = 0.87, \phi = 0.01)$	$(\delta = 0.95, \phi = 0.01)$	$(\delta = 0.91, \phi = 0.01)$	$(\delta = 0.87, \phi = 0.01)$
Over All Games:	$(\delta = 0.91, \phi = 0.06)$	$(\delta = 0.91, \phi = 0.06)$	$(\delta = 0.91, \phi = 0.06)$	$(\delta = 0.91, \phi = 0.06)$	$(\delta = 0.91, \phi = 0.06)$
ϵ-Greedy					
Individual Fits:	$(\delta = 0.73, \phi = 0.03)$	$(\delta = 0.97, \phi = 0.58)$	$(\delta = 0.85, \phi = 0.01)$	$(\delta = 0.97, \phi = 0.42)$	$(\delta = 0.99, \phi = 0.35)$
Cross-Validation:	$(\delta = 0.98, \phi = 0.01)$	$(\delta = 0.9, \phi = 0.01)$	$(\delta = 0.92, \phi = 0.01)$	$(\delta = 0.9, \phi = 0.01)$	$(\delta = 0.87, \phi = 0.01)$
Over All Games:	$(\delta = 0.9, \phi = 0.04)$	$(\delta = 0.9, \phi = 0.04)$	$(\delta = 0.9, \phi = 0.04)$	$(\delta = 0.9, \phi = 0.04)$	$(\delta = 0.9, \phi = 0.04)$
Softmax					
Individual Fits:	$(\delta = 0.92, \phi = 0.66)$	$(\delta = 0.88, \phi = 0.37)$	$(\delta = 0.65, \phi = 0.27)$	$(\delta = 0.6, \phi = 0.8)$	$(\delta = 1, \phi = 0.1)$
Cross-Validation:	$(\delta = 0.89, \phi = 0.01)$	$(\delta = 0.89, \phi = 0.01)$	$(\delta = 0.97, \phi = 0.01)$	$(\delta = 0.89, \phi = 0.01)$	$(\delta = 0.88, \phi = 0.01)$
Over All Games:	$(\delta = 0.89, \phi = 0.07)$	$(\delta = 0.89, \phi = 0.07)$	$(\delta = 0.89, \phi = 0.07)$	$(\delta = 0.89, \phi = 0.07)$	$(\delta = 0.89, \phi = 0.07)$

References

[AM64] R J. Aumann and M Maschler, *The bargaining set for cooperative games*, Advances in Game Theory (M Dresher, L S. Shapley, and A W. Tucker, eds.), Princeton University Press, 1964.

[CH98] Colin F. Camerer and Teck-Hua Ho, *Experience-weighted attraction learning in coordination games: Probability rules, heterogeneity, and time-variation*, Journal of Mathematical Psychology **42** (1998), 305–326.

[CH99] ———, *Experience-weighted attraction learning in normal form games*, Econometrica **67** (1999), no. 4, 827–974.

[CK04] Alex K. Chavez and Steven O. Kimbrough, *A model of human behavior in coalition formation games*, Proceedings of the 4th Annual International Conference on Cognitive Modeling, July 2004.

[DKL96] Garett O. Dworman, Steven O. Kimbrough, and James D. Laing, *On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game*, Journal of Management Information Systems **12** (1995–96), no. 3, 97–125.

[DKL95a] ———, *Bargaining in a three-agent coalitions game: An application of genetic programming*, Working Notes: AAAI–95 Fall Symposium Series, Genetic Programming (Boston, MA, November 10-12, 1995), AAAI, 1995, pp. 9–16.

[DKL95b] ———, *On automated discovery of models using genetic programming in game–theoretic contexts*, Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences, Volume III: Information Systems: Decision Support and Knowledge-Based Systems (Los Alamitos, CA) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1995, pp. 428–438.

[DKL96] ———, *Bargaining by artificial agents in two coalition games: A study in genetic programming for electronic commerce*, Genetic Programming 1996: Proceedings of the First Annual Genetic Programming Conference, July 28-31, 1996, Stanford University (John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, eds.), The MIT Press, 1996, pp. 54–62.

[ER98] Ido Erev and Alvin E. Roth, *Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria*, The American Economic Review **88** (1998), no. 4, 848–881.

[FL98] Drew Fudenberg and David K. Levine, *The theory of learning in games*, The MIT Press, Cambridge, MA, 1998.

[Gal90] Charles R. Gallistel, *The organization of learning*, The MIT Press, Cambridge, MA, 1990.

[GS93] Dhananjay K. Gode and Shyam Sunder, *Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality*, Journal of Political Economy **101** (1993), no. 1, 119–137.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KR74] James P. Kahan and Amnon Rapoport, *Test of the bargaining set and kernel models in three-person games*, Game Theory as a Theory of Conflict Resolution (Anatol Rapoport, ed.), D. Reidel, Dordrecht, The Netherlands, 1974, pp. 119–160.

[KR84] ———, *Theories of coalition formation*, Lawrence Earlbaum Associates, Hillsdale, NJ, 1984.

[LR57] R. Duncan Luce and Howard Raiffa, *Games and decisions*, John Wiley, New York, NY, 1957, Reprinted by Dover Books, 1989.

[MF02] Michael W. Macy and Andreas Flache, *Learning dynamics in social dilemmas*, Proceedings of the National Academy of Science (PNAS) **99** (2002), no. suppl. 3, 7229–7236.

[RE95] Alvin E. Roth and Ido Erev, *Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term*, Games and Economic Behavior **8** (1995), 164–212.

[SB98] Richar S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, The MIT Press, Cambridge, MA, 1998.

[SC95] T. Sandholm and R. Crites, *Multiagent reinforcement learning in iterated prisoner's dilemma*, Biosystems **37** (1995), 147–166, Special Issue on the Prisoner's Dilemma.

[SV99] R. Sarin and F. Vahid, *Payoff assessments without probabilities: A simple dynamic model of choice*, Games and Economic Behavior **28** (1999), 294–309.

[SV01] ———, *Predicting how people play games*, Games and Economic Behavior **34** (2001), 104–122.

[Uhl90] Gerald R. Uhlich, *Descriptive theories of bargaining*, Springer-Verlag, Berlin, Germany, 1990.

On Learning Negotiation Strategies by Artificial Adaptive Agents in Environments of Incomplete Information

Jim R. Oliver

INSEAD, France,
jroliver@ebay.com

Abstract. Automated negotiation by artificial adaptive agents (AAAs) holds great promise for electronic commerce, but non-trivial, practical issues remain. Published studies of AAA learning of negotiation strategies have been based on artificial environments that include complete payoff information for both sides of the bargaining table. This is not realistic in applied contexts. Without loss of generality, we consider the case of a seller who knows its own preferences over negotiation outcomes but will have limited information about the private values of each customer. We propose a learning environment that takes advantage of partial information likely to be available to the vendor. General strategies are learned for a group of similar customers – a market segment – through a simulation approach and a genetic learning algorithm. In addition, we systematically further relax constraints on the opponent's preferences to further explore AAA learning in incomplete information environments.

1 Introduction

Researchers have demonstrated that artificial adaptive agents (AAAs) can learn effective negotiation strategies. But the experimental environments in which these agents learn are also artificial and depart from what is plausible in natural settings. In particular, the environments completely specify the payoff structure for the opposing sides, but in applied contexts each side keeps information private or even misrepresents it. One side never completely knows the other. Still, we have some hunches about our opponents. Wouldn't it be nice if there were a way to take advantage of what is known about our opponent and incorporate it systematically and effectively into a learning environment for AAAs? This paper examines the potential of and perils of incomplete information and the impact on AAA performance.

1.1 The Challenge of Negotiation

Bargaining and negotiation always have, and always will, play a critical role in commerce. Even within a firm, allocation decisions usually have a negotiation component. Despite its importance and prevalence throughout history,

human negotiation performance falls significantly short of optimal. Experiments, field studies, and our common experience demonstrate that even in simple negotiations people often reach sub-optimal agreements, thereby "leaving money on the table" [Cam90]. Economic efficiency would be greatly increased if the cost of negotiation could be decreased or superior agreements could be effected. Computer science and information systems researchers have approached this problem in two ways. One stream of research has sought to provide tools that support human negotiators (e.g. [RS97]), while another stream seeks to build systems that can negotiate by themselves. Our interest here is in the latter approach, specifically systems of Artificial Adaptive Agents (AAAs) based on evolutionary computation.

The essence of negotiation is two or more parties trying to arrive at a single agreement from a set – often large – of potential agreements. Part of the challenge and the opportunity of negotiation arises from differences between the parties as to how they value possible outcomes.

Negotiation can be viewed as a *search* process. Optimization is also often thought of as a search process, but it differs in important ways from negotiation search. Optimization seeks the maximum or minimum of a known (computable), static, search space. In negotiation, each side has private information, and neither knows the other's exact utility function. The situation is further complicated because both sides have incentive to misrepresent their preferences. Thus, each party has differing views of the search space, which change over time as the bargaining session unfolds. Finding an optimal agreement in this dynamic environment is extremely challenging because both sides are in competition, but must jointly search the space of possible agreements.

Despite the challenging nature of negotiation, researchers pursuing automated negotiation have had provocative successes. A multitude of approaches have been taken, and the results have been reviewed extensively elsewhere (e.g. [Bic01]; [Far98]); thus we will only provide brief background. [CSLC01], [TWL00], and [Oli96a] used genetic algorithms [Gol89] to evolve agents that can negotiate in complex, large, multi-dimensional search spaces. [Oli96a] used simple strategies that capture common decision making processes, but lack the expressive power to capture and react directly to a particular dynamic bargaining path. [TWL00] extended the work by using Finite State Machines (FSMs) to represent strategies. Another stream of research is represented by [DKL96] who use genetic programming techniques to evolve agents that can play coalition games.

1.2 Two Problems: Learning Curves and Knowing Your Opponent

The promise of Artificial Adaptive Agents learning to negotiate is exciting, but many practical problems remain to be overcome before the approach is viable for commercial transactions between firms. We turn our attention to just one of these difficulties, a specific applied problem that is important and

that also was the original motivation for the investigations reported here. Consider the potential situation of a vendor wishing to deploy a set of AAAs that will be able to negotiate with its customers. These customers could be human buyers or perhaps automated agents themselves. By focusing on just this particular situation, we get around a number of problems such as choosing a common negotiation protocol, a shared ontology, and commitments, because in this situation the vendor defines the rules of the game, and it is incumbent upon the customer to comply[1].

In this simplified case, we are left with only the fundamental challenge: how do the AAAs learn to negotiate the myriad transactions in the first place? How do they learn to implicitly or explicitly "understand" the customers? We might assume that the vendor would wish to leverage the previously mentioned studies demonstrating that artificial agents, directed by evolutionary algorithms, can effectively learn negotiation strategies. However, the published approaches have two characteristics that, when taken together, create a serious practical problem. The first problem we call the *learning curve* problem. Although evolutionary learning is effective in many ways, it is not fash enough for all situations; genetic algorithm (GA) and genetic programming (GP) based learning requires hundreds or even thousands of trials to learn effective strategies. The learning curve, or rate at which the agent improves, is reasonable when compared to the size of the problems, but as a practical matter this characteristic prevents a vendor from training AAAs "on the job," i.e., through real customer interactions. Researchers have avoided this problem by pitting computer agents against each other. Although computer cycles, by relevant measures, are cheap and fast, it is here that the other half of the problem emerges. To play against the other side requires a model of the other side. That is to say we need a complete environment for the AAAs that covers both sides of the negotiating table. We argue that our hypothetical vendor is likely to have some ideas about its customer base but not likely to have complete access to the subtle preferences its many customers might have. We call this second problem the *know your opponent* problem.

To summarize, effective AAA learning of negotiation strategies requires either a real opponent willing to do thousands of trials or models of the opponent. We explore one approach to building those models, one that leverages knowledge likely to be available in applied settings but also likely to be incomplete. We then generalize to other forms of incomplete information with that aim of shedding light on the potential and the perils of incomplete models on AAA performance.

[1] For widespread adoption, such a vendor dictated approach is inadequate, but it is not unreasonable that there are situations in which a vendor could take the lead in such a manner.

2 Overview of the Approach

The basic approach is as follows. We assume that a vendor can describe some characteristics of the preferences for a group of customers, interpreted as a particular customer segment. This description will be the basis of an incomplete specification of that segment. We aim, in essence, to exploit approximate models, the details of which will be made clear in a later section. From the basic characteristics of a market segment, multiple customers are simulated, and the AAAs of the vendor learn to negotiate with these many instances of similar customers. The vendor agent learns a general strategy for the segment with the hope that new but similar customers can be effectively dealt with. This type of general strategy learning is related to [Oli97b].

3 AAA Platform

We now describe the system that is used in these investigations, addressing representation of the negotiation space, modeling of the participants, and system operation. In the spirit of the motivating example, i.e., a vendor training AAAs to negotiate with a customer segment, we use the terms vendor and customer to refer to the different parties and their different models. A numerical example is provided at the end of the section for clarification.

Evolutionary Learning. The heart of the system is the genetic algorithm that drives the learning process for the artificial agents. The details of the algorithm will not be presented; the GA employed was fairly simple, "vanilla" form [Gol89]. Agents learn from their experience bargaining as follows. After initializing strategies as described below, a randomly chosen agent, either a vendor agent or a customer agent, begins a bargaining round with an offer. The receiving agent evaluates the offer and either accepts it or makes a counter offer. The exception to this occurs when a system-specified maximum number of offers has been reached, in which case bargaining is terminated and each agent received a default payoff of zero. This bargaining cycle continues for a system specified number of rounds, enough to test the effectiveness of the individual chromosomes/strategies in the population. The GA is run and this sequence makes one generation. The process continues for a system specified number of generations, enough that the population becomes effective at bargaining.

Strategies. Both sides of the negotiation use the same strategy structure in our experiments. We use simple, sequential, threshold decision rules as the basis of negotiation strategies, as used by [Oli96a]. An example of this type of strategy for a vendor agent might be as follows. Initially, accept any customer offer whose utility is greater than a threshold T_1. If the customer's offer does not meet that threshold, then make a counter offer. If the customer does not accept this offer, and another counter offer is made, the purchasing agent will compare this to another, typically different, threshold, T_2. Again, if the

threshold is not met, a counter offer could be made. This type of decision rule could be called an adaptive satisficing strategy [Alb88].

In the future, we could consider richer representations of strategies, such as finite state machines[2] or genetic programming,[3] which allows the strategic structure and complexity to be chosen endogenously.

Initializing Strategies. The agents are not endowed with any initial strategic knowledge about offers to make or how to react to offers. Following typical practice, the initial strategies, for both the vendor and the customers, are randomly created, i.e., random acceptance thresholds and random offers. In the early phases of the learning process, low value offers are routinely made and also accepted, but these inferior behaviors are rapidly eliminated through the operation of the GA.

The Bargaining Space: Negotiable Issues and Alternatives. Multi-ple-issue negotiations are characterized by a set of issues, each of which has two or more alternatives. This representation generates a multi-dimensional space of possible agreements. An offer or potential agreement is a vector in this space.

Vendor Utilities. Utilities are stored as simple additive value models. Each issue has a weight and each possibility for each issue has a numerical value. The value of an option is simply the weighted sum. The value, V, of a particular option $X = (x_1, x_2, ..., x_n)$ is,

$$V(X) = \sum_i w_i v_i(x_i), \tag{1}$$

where, x_i is the level of alternative X on the ith issue, v_i is the part-worth function for the alternatives for issue i, and w_i is a weighting factor that may or may not be necessary depending on the scaling of the v_i terms. The simple, additive preference models in use by the current system assume each alternative is preferentially independent. Extensions to accommodate specific interactions could be easily made.

Table 1 provides a numerical example that will make the value function representation clearer. The example is that of a computer manufacturer or vendor that offers products with a static physical design and microprocessor type, but the customer is allowed to configure and negotiate certain attributes such as price, processor speed, memory, and delivery schedule. In the table, processor speed is the most important issue and the most preferred option, 3.0 GHz, contributes .35 towards the overall value. The table shows the weighted values, which are the product of the weight for the issue times the value of the option. In this case, the weights for (price, processor, memory, delivery) are (.3, .35, .25, .1). Looking at just price alone, the options (1000, 1250, 1500, 1750) have individual values of (1.0, .67, .33, 0). The product of the weight for price, .30, times the previous values yields the weighted values in the table

[2] [TWL00]
[3] [DKL96]

of (.30, .20, .10, 0). The value of any particular option is straightforward to calculate. For example, the option (price = 1500, processor = 2.0 GHz, memory = 128 MB, delivery = 15 days) is .10 + .20 + 0 + .03 = .33.

Table 1. Example additive value function

Issues	Alternatives	Weighted Values
Price ($)	1000	.30
	1250	.20
	1500	.10
	1750	0
Processor (GHz)	1.5	0
	2.0	.20
	2.5	.25
	3.0	.35
Memory (Mbytes)	128	0
	256	.15
	384	.20
	512	.25
Delivery (Days)	2	.10
	5	.08
	15	.03
	30	0

All component values are scaled so that the maximally valued option has a value of 1.0 and the worst has a value of 0.0. In an applied setting value functions can be linearly transformed to another more convenient scale for display on a user interface, but this would only be for user convenience and interpretation.

Modeling Customer Segments. Although we assume that the vendor's exact, cardinal preferences are knowable, i.e., an additive value function can be elicited, this is not the case with customer preference information. Although detailed preference information is unavailable to the vendor, at least some constraints on those preferences are likely to be knowable. One interpretation of the approximate information might be that it represents a customer type or a particular market segment. For the investigations reported here, we specifically assume that the vendor has access to some ordinal preference information, i.e., the vendor knows how the customer segment ranks various possibilities.

To make this notion clearer, consider the example personal computer manufacturer. While the vendor might not know customer preferences to the extent shown in Table 1 – knowing the exact value of a 1.5 GHz processor versus a 1.8 GHz processor – the vendor *is* likely to know which attributes are more important. For example, consider the case of graphic designers working for the

government. This group of customers might value processor speed first, for complex rendering, followed by memory and price, and finally delivery, which in this case might be least important because of advance planning. Ranking the preferences within an attribute is straightforward: more memory is desired over less memory, faster is better than slower, and so forth. Continuing this process for the remaining attributes constitutes the complete ordinal preference ranking.

To run learning sessions, cardinal value functions are required and these are generated from a specified ordinal preference ranking. In our experiments, individual customer instances are created randomly, each fitting the ordinal constraints, and in this manner, we simulate customers within the segment.

4 Structure of the Bargaining Space

As our investigations of incomplete information will be based on ordinal preference information, it is important to note the extent to which this constrains what the bargaining space "looks like." Figure pairs 1 and 2 show how two instances of the same ordinal preferences create strikingly different bargaining spaces, ranging from sharply clustered agreement points (Figure 1) to scattered bargaining points that create an outward-bowed frontier (Figure 2). The figures are for the case of 4 issues, 4 alternatives for each issue, and continuous value functions scaled to the interval [0,1] similar to Table 1. For a careful analysis of how individual value functions, for two issues and two agents, interact to create dramatically different bargaining spaces, the reader can refer to [Mum91].

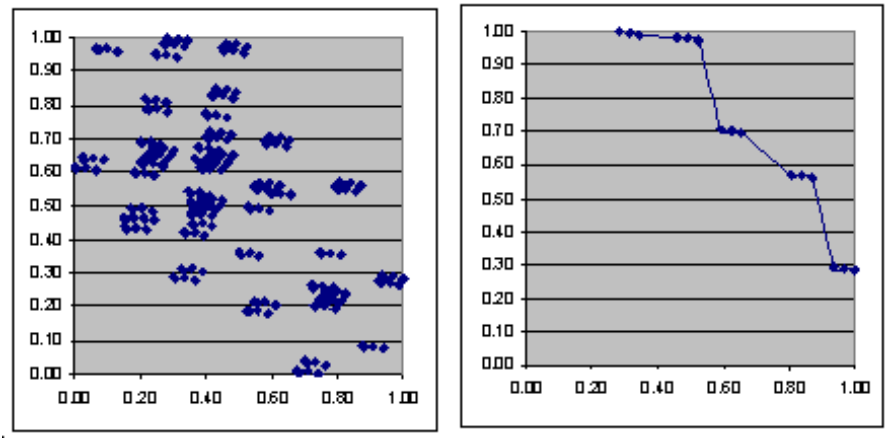


Fig. 1. Example bargaining space and corresponding pareto frontier

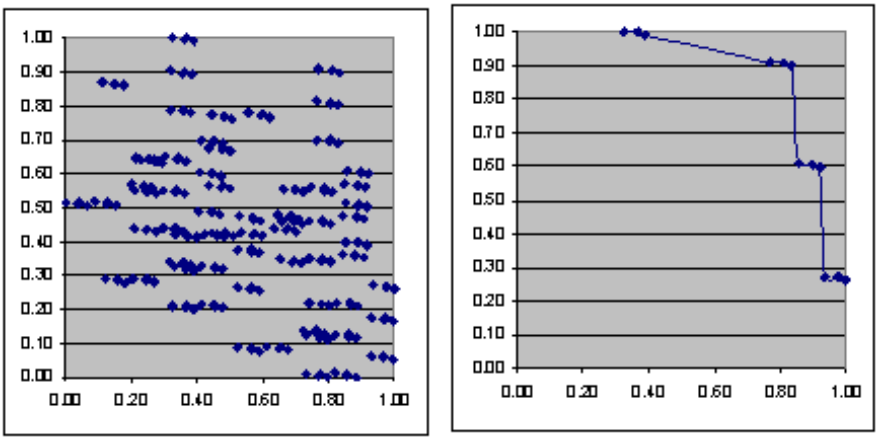


Fig. 2. Bargaining space related to Figure 1 and corresponding Pareto frontier

The above example bargaining spaces are more general than our applied example. We assume the vendor knows its value function precisely, but the customer preferences are less precisely known. Figure 3 shows an example of the variation in the bargaining space under comparatively minor changes in the cardinal value function for the customer only. Tables 2 and 3 show that the cardinal values for the vendor are unchanged, and the customer weights are also unchanged, but the customer's relative weights for the alternatives within each issue vary between the two tables. The ordinal ranking of the alternatives is the same in the two tables.

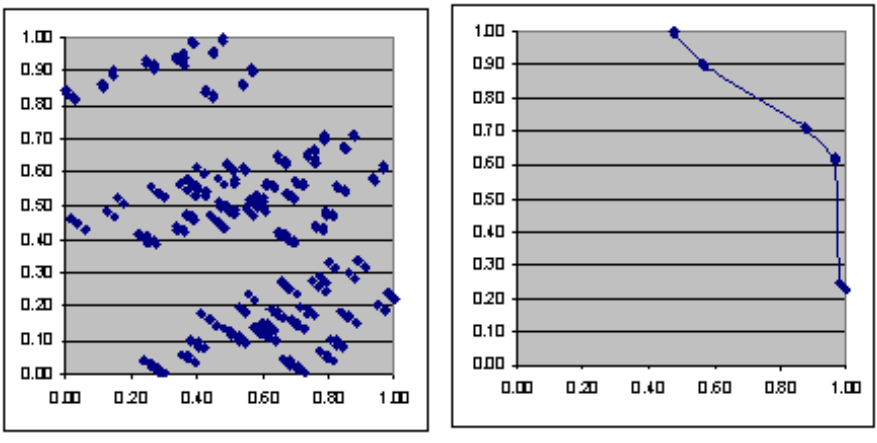


Fig. 3. Bargaining space and Pareto frontier for Table 2

Table 2. Vendor and customer value functions for figure 3. Bold numbers indicate a most preferred alternative.

Vendor						**Customer**				
	Alternative						Alternative			
Issue	1	2	3	4		Issue	1	2	3	4
1	**0.424**	0.000	0.244	0.334		1	0.006	0.000	0.092	**0.100**
2	**0.034**	0.014	0.003	0.000		2	0.000	0.017	0.385	**0.395**
3	0.000	**0.400**	0.244	0.224		3	**0.430**	0.145	0.000	0.007
4	0.024	**0.142**	0.000	0.113		4	0.000	**0.075**	0.016	0.036

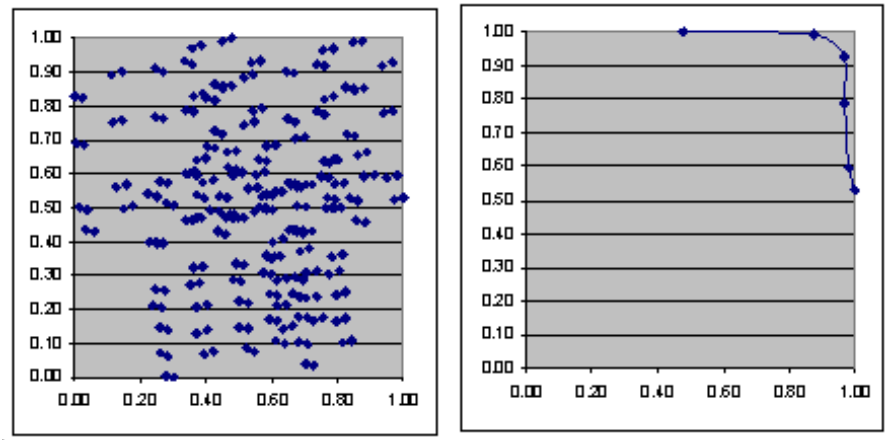


Fig. 4. Bargaining space and Pareto frontier for Table 3

Table 3. Vendor and customer value functions for figure 4. Bold numbers indicate a most preferred alternative.

Vendor						**Customer**				
	Alternative						Alternative			
Issue	1	2	3	4		Issue	1	2	3	4
1	**0.424**	0.000	0.244	0.334		1	0.034	0.000	0.077	**0.100**
2	**0.034**	0.014	0.003	0.000		2	0.000	0.065	0.254	**0.395**
3	0.000	**0.400**	0.244	0.224		3	**0.430**	0.423	0.000	0.140
4	0.024	**0.142**	0.000	0.113		4	0.000	**0.075**	0.006	0.068

5 Experimental Testing and Results

We test the learning capabilities of the AAAs through detailed investigations of increasingly relaxed constraints on customer preferences. The particular parameters of the GA were selected based on preliminary testing to determine values that would reliable create high performing agents. The key parameter

values are as follows. For recombination, the GA parameters are P(crossover) = 0.5 and P(mutation) = 0.025. The total number of bargaining sessions in an experiment depends on the number of bargaining sessions for each generation, population size, and number of generations; various combinations of these that extended out to be tens of thousands of bargaining sessions effected consistently strong learning by the agents.

5.1 Experimental Design

The performance of AAAs with fully specified opposing agents forms the baseline of performance against which we compare the AAA performance in the cases of incomplete information. In the baseline case, cardinal preferences are specified for both issues and for alternatives on each issue, i.e., fully specified value functions. This is the case in which individual customer preferences are known precisely and is the best case for learning high-performance negotiating strategies.

In successive cases, the constraints on customer preferences are relaxed in systematic ways. Table 4 summarizes each case. Case 1 specifies cardinal preferences for issues, but ordinal preferences on alternatives. This corresponds to knowing the exact relative weight of each issue – price versus processor speed, for example – but only knowing that higher price is preferred to lower price without knowing quantitatively by how much. Case 2 switches these: ordinal issue preference and cardinal preferences over alternatives. This corresponds, in our example, to knowing precisely the *relative* value of each price point versus the others, each processor speed versus the others, and so forth, but only knowing from a ranking viewpoint that processor speed is more important that price. Cases 3 and 4 relax the constraints further to the point that might be unusual for applied contexts. Case 3 places no constraints on the importance of alternatives within each issue, but preserves among those in the customer group a consistent preference order on the issues themselves. No preferences over alternatives would be unrealistic in the case of price but not necessarily in the case of an issue such as color. Case 4 places the opposite constraints.

The final case, number 5, is meant to approximate more natural conditions of a vendor/customer interaction. The key difference in this case is that in the ordering of preferences over alternatives, we force the vendor and customer to have perfectly opposing interests. For example, the vendor would like to sell the lowest cost computer for the highest price and with the longest lead-time. In contrast, the customer would like the highest performance computer at the best price as soon as possible. The preferences among the issues themselves, however, are not constrained and, in general, are not the same. The difference in preferences creates a trade-off so that there is a win-win opportunity to take advantage of. For example, the vendor might be exceptionally busy and would like to stretch out lead-time, whereas the customer segment generally plans ahead and does not need computers delivered right away.

Table 4. Test cases of incomplete information

Case	IssuePreference	Alternative Preference
Baseline	Cardinal	Cardinal
1	Cardinal	Ordinal
2	Ordinal	Cardinal
3	Ordinal	None
4	None	Ordinal
5 (Opposing)	Cardinal	Ordinal

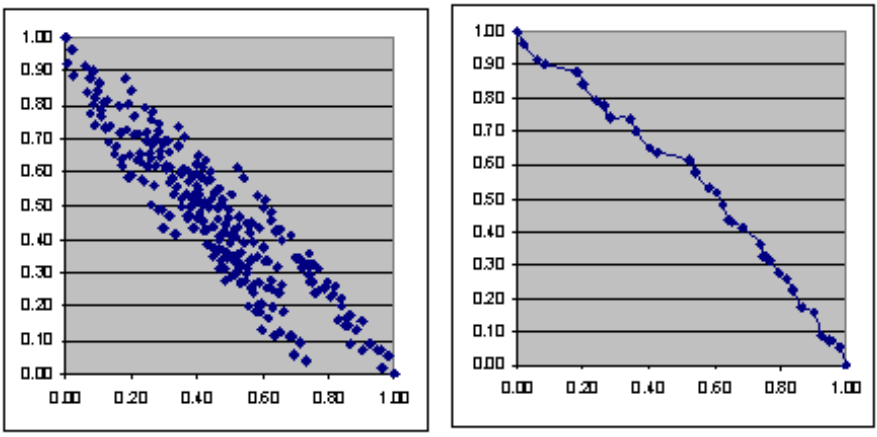


Fig. 5. Bargaining space for opposing interests and corresponding Pareto frontier.

Case 5 creates a far more distributive bargaining space, with only modest integrative opportunity as compared to the other cases. Figure 5 shows an example of the bargaining space for this case. The primary opportunities for gain are at the expense of the other player, i.e., by moving among points in a northwest or southeast direction on the figure. Unlike the other cases and example bargaining spaces, the extreme payoffs in this case to (vendor, customer) are (1,0) and (0,1). Points northeast (above and to the right) of other points represent improvements in payoffs to both parties, but these opportunities have a smaller improvement than those shown in Figures 1 through 4. Not all the possibilities for case 5 have bargaining spaces that are as compressed or flat as is shown in this example; the compression is because in figure 5 the opponents both place the greatest value on the same issue. When this is not the case, then the integrative opportunity is greater.

Each test case represents a broad market segment, and for each we test the performance of the strategies learned for the segment to the performance of more tailored strategies unique to each customer – strategies which could

be learned if customer preferences were completely known. The experimental procedure is as follows. Value function generation is a two-step process. First, random sets of ordinal preferences, covering both the issues and the alternatives for each issue, are generated. Two sets of preferences are generated, one for each side of the bargaining table. Second, cardinal preferences are generated that fit the ordinal preferences. For the customer side, portions of this process, as appropriate, are repeated m times to generate m value functions corresponding to m customers fitting the structure imposed by the test case. The single vendor that is paired with each of the m unique customers creates a unique bargaining space. We calculate and save the average value (to each player separately) of all the points in the space, the average value of each point on the frontier, and the average distance of each point to the frontier. These statistics summarize the bargaining space. In the experiments, a value of m=20 was adequate for consistent results, but no investigation was performed to understand how this value interacted with or depended on other parameters.

Each test case proceeds in two phases. First, the vendor agent trains with a single customer agent and learns a unique strategy for that customer. Next, a fresh copy of the vendor agent competes against a new customer. This continues for all m customers simulated for the segment. The vendor agents are all identical in that they have exactly the same preferences. Data summarizing the bargaining performance is saved and provides the baseline benchmark in which the vendor is allowed to evolve specific strategies for each customer.

In the second phase of each test case, a new instantiation of the vendor agent trains against "experienced" customer agents and evolves general strategies for the segment. First the trained customer agents from the previous phase are split into two groups (equal size), one for training and a hold-out sample for evaluation. The vendor agent is matched with one of the saved customers from the training group (specifying both strategies and value functions); these agents use strategies that correspond to those an experienced customer agent might use. A series of bargaining rounds occurs, following which the GA is run, *for the vendor only.* In the next generation, the vendor is matched with a new (i.e., expectedly different) saved customer, drawn from the training pool. This process continues for the same number of generations as in the first phase. At this point, the vendor agent has learned a "general" strategy for the customers in the training segment. An evaluation step captures the performance of the vendor agent by pairing it with each customer agent from the holdout sample. The results of a standard bargaining session are saved for each interaction.

5.2 Results

We test agent learning in two ways. First, we statistically compare the payoffs earned by each agent, after their bargaining experience, to the average payoff

all the agreements in the bargaining space, i.e., the expected payoff of a random offer. Second, we compare the earned versus random payoffs in terms of their distance to the Pareto frontier. Being closer to the Pareto frontier is a sign of achieving integrative agreements that take advantage of the different agent preferences in a win-win manner.

t-tests are used to test whether the bargaining outcomes of the trained agents differ from random points in the bargaining space. Table 5 summarizes the means and standard deviations for bargaining outcomes in terms of payoff to each agent; in the table, Player 1 is the vendor and Player 2 is the customer. For the first test, of payoffs to each agent, the results were highly significant for all cases ($p < .0002$ or much better). Based on the column headings in Table 5, the first test compares the raw data (not shown) for AllPoints Ave 1 to Baseline Ave 1 and General Ave 1, and the same comparison is done for player 2. The second tests, of nearness to frontier, were also highly significant ($p < .02$ or much better). Table 6 summarizes the means and standard deviations for bargaining outcomes in terms of nearness to frontier.

We now address the question of whether the specialized vendor agents perform better than the generalized agents that learn for an entire segment. Looking at Table 5, we note that in Case 1, the mean payoff to Player 1 (the vendor) is .773 in the baseline test and .695 in the general strategy test, suggesting that the specialized agents that learn individual strategies for each customer are more effective negotiators. To verify this, we again perform t-tests that compare the payoffs for each agent in each case. In Table 5, we compare the raw data (not shown) for Baseline Ave 1 to General Ave 1 and Baseline Ave 2 to General Ave 2. All are highly significant (p=1x10^{-7} or better) indicating that the varying performance penalty for the cases of incomplete information is real. The significance also extends to the customer side, Player 2; these "experienced" players suffer from the bargaining disadvantage of the vendor.

Finally we explore which cases of incomplete information are the most "difficult" for the agents. As a proxy for difficulty we measure the penalty that the dyad experiences in each case, that is the payoff shortfall to each player when the vendor agent must learn general strategies for the entire segment. Table 7 shows the average penalties and the range for a 95% confidence interval. The smallest penalty, certainly from the vendor's viewpoint, is case 2, and the greatest penalty is case 3. Case 2 deviates from cardinal utilities only for the issues, whereas Case 3 removes all restrictions on the alternatives, of which there are collectively many more than issues. This result is not terribly surprising on the surface, given the likely impact on the bargaining space, but more research would be required to fully understand the effect, as well as the less obvious result that the penalty for the customer is very similar in three of the cases, despite differences in penalties for the vendor.

Table 5. Mean and standard deviation of payoffs for each test case. In the table, the vendor agent is coded with a 1 and the customer with a 2.

	AllPoints	**AllPoints**	**Frontier**	**Frontier**	**Baseline**	**Baseline**	**General**	**General**
CASE	**Ave1**	**Ave2**	**Ave 1**	**Ave 2**	**Ave 1**	**Ave 2**	**Ave 1**	**Ave 2**
1 mean	.495	.502	.797	.793	.773	.778	.695	.704
std dev	.061	.021	.103	.089	.086	.080	.105	.109
2 mean	.499	.497	.815	.807	.790	.786	.725	.712
std dev	.057	.019	.095	.080	.073	.068	.099	.102
3 mean	.498	.501	.791	.790	.770	.770	.676	.678
std dev	.056	.019	.043	.031	.055	.043	.087	.069
4 mean	.501	.505	.790	.804	.770	.779	.692	.708
std dev	.052	.017	.098	.073	.076	.068	.097	.099
5 mean	.499	.503	.609	.617	.609	.625	.539	.549
std dev	.057	.019	.059	.045	.079	.071	.089	.100

Table 6. Mean and standard deviations of distance to frontier of the average point in the bargaining space and the bargaining outcomes achieved by trained agents in each test case.

	AllPoints Ave	**Baseline Ave**	**General Ave**
CASE	**Dist to Frontier**	**Dist to Frontier**	**Dist to Frontier**
1 mean	.080	.015	.033
std dev	.021	.005	.017
2 mean	.085	.016	.033
std dev	.017	.005	.016
3 mean	.079	.015	.037
std dev	.007	.005	.012
4 mean	.080	.016	.033
std dev	.018	.005	.014
5 mean	.034	.012	.030
std dev	.009	.004	.014

Table 7. Payoff penalties when vendor agents learn general strategies for a segment

Case	**Baseline**		**Penalty**	
	Vendor	**Customer**	**Vendor**	**Customer**
1	0.773	0.778	0.078±.013	0.074±.018
2	0.790	0.786	0.065±.016	0.074±.016
3	0.770	0.770	0.094±.012	0.091±.013
4	0.770	0.779	0.078±.011	0.070±.015
5	0.609	0.625	0.070±.009	0.076±.017

6 Conclusion

The results of these experiments show that agents can learn strategies in environments of incomplete information, but there is a penalty as compared to the benchmark of complete information. The nature of the incomplete information that we explored was very specific, motivated in part by applied considerations. The extent to which our models of customer segments, simulated from ordinal preference data, are reasonable and appropriate in applied contexts is an empirical question that deserves more attention. One can imagine other models of customer segments, such as creating uniform intervals around each point value in a cardinal value function, to name just one. This interval approach is reminiscent of sensitivity analysis, and a related research question is: given an AAA negotiation strategy, over what range of opponent preferences is that strategy effective?

These experiments provide only initial insights into agent performance in incomplete information environments. The agents live in the bargaining spaces created by the modeling and simulation techniques. More work is

needed to understand the links between (1) the impact of incomplete information to the structure of the bargaining spaces, and (2) the interplay between the structure of the bargaining space and agent learning. In our experiments, we combined these effects because the overall effect was what we wished to research. Additional simulations or even analytical work might provide key insights into the component effects.

Previous studies have compared the negotiation performance of AAAs with that of humans. An important follow-on test of the simulation approach described here would be to have the trained agents negotiate directly with human agents who are given the appropriate preference information. This would be the best test of the system prior to deploying in a natural environment.

Human testing, as described, is valuable for validation, but we point out, in closing, that it is not a practical approach for investigating the questions posed in this research project. Negotiation outcomes, by both human and artificial agents, are very noisy. Teasing out the impact of small variations in structure requires a great deal of data, enough that it would be impractical to obtain from human experiments but it is quite easy to get through computational approaches.

References

[Alb88] W. Albers, *Aspirations and aspiration adjustment in location games*, 1988.

[Bic01] M. Bichler, *The future of e-markets: Multi-dimensional market mechanisms*, Cambridge University Press, Cambridge, United Kingdom, 2001.

[Cam90] Colin F. Camerer, *Behavioral game theory*, Insights in Decision Making: A Tribute to Hillel J. Einhorn (R.M. Hogarth, ed.), Univ. of Chicago Press: Chicago, IL, 1990, pp. 311–336.

[CSLC01] S. Choi, P. M. Samuel, J. Liu, and S. P. Chan, *A genetic agent-based negotiation system*, Computer Networks **37** (2001), no. 2, 195–204.

[DKL96] Garett O. Dworman, Steven O. Kimbrough, and James D. Laing, *On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game*, Journal of Management Information Systems **12** (1995–96), no. 3, 97–125.

[Far98] P. Faratin, *Negotiation among groups of autonomous computational agents. thesis*, Ph.D. thesis, Department of Electronic Engineering, Queen Mary and Westfield College, University of London, 1998.

[Gol89] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley: Reading, MA, 1989.

[Mum91] J. Mumpower, *The judgment policies of negotiators and the structure of negotiation problems*, Management Science **37** (1991), no. 10, 1304 – 1324.

[Oli96] Jim R. Oliver, *A machine learning approach to automated negotiation and prospects for electronic commerce*, Journal of Management Information Systems **13** (1996), no. 3, 83 – 112.

[Oli97] ———, *Artificial agents learn policies for multi-issue negotiation*, International Journal of Electronic Commerce **1** (1997), no. 4, 49 – 88.

[RS97] A. Rangaswamy and R. Shell, *Using computers to realize joint gains in negotiations: Toward and electronic bargaining table*, Management Science **43** (1997), no. 8, 1147.

[TWL00] M. Tu, E. Wolff, and W. Lamersdorf, *Genetic algorithms for automated negotiations: a FSM-based application approach*, Proceedings of 11th International Conference on Database and Expert Systems (DEXA 2000), 2000.

A Note on Strategic Learning in Policy Space

Steven O. Kimbrough[1], Ming Lu[2], and Ann Kuo[3]

[1] University of Pennsylvania, Philadelphia, PA, USA,
kimbrough@wharton.upenn.edu
[2] University of Pennsylvania, Philadelphia, PA, USA,
milu@wharton.upenn.edu
[3] University of Pennsylvania, Philadelphia, PA, USA,
kuo2_99@yahoo.com

Abstract. We report on a series of computational experiments with artificial agents learning in the context of games. Two kinds of learning are investigated: (1) a simple form of associative learning, called Q-learning, which occurs in state space, and (2) a simple form of learning, which we introduce here, that occurs in policy space. We compare the two methods on a number of repeated 2×2 games. We conclude that learning in policy space is an effective and promising method for learning in games.

1 Introduction

In what follows we report on a series of computational experiments with artificial agents learning in the context of games (situations of interdependent decision making). We discuss two kinds of learning. The first is a form of simple associative learning, called Q-learning (a form of reinforcement learning) in the machine learning literature.[1] This sort of reinforcement learning by agents in games has been investigated previously by a number of researchers.[2] We introduce, in our discussion of the second series of experiments, a new variety of reinforcement learning in the context of games. This kind of learning appears to be both very effective from the point of view of the agents and cognitively plausible.

We begin by providing, in the next two sections, some essential context and background.

2 Background: Games and Decisions

Decision contexts faced by agents may be distinguished into those that are and those that are not *strategic*. In *non-strategic contexts,* decisions by other agents do not need to be taken into account. When, for example, one decides

[1] Cf., [KLM96,SB98].
[2] E.g., [Cam03], [KL03], [KL04], and [SC95].

whether to dress for rain on a given day, what matters most is whether it will in fact rain.[3] This fact, whichever way it turns out, is not in any way dependent on the decisions of another agent. Nature, no matter how we joke otherwise, does not care about and does not have interests in whether we get wet or not. Similarly, the decisions made by an animal in navigating towards a goal do not (at least in many cases) need to take into account decisions by other animals. Constructing and exploiting a map gets the agent/animal home, not, e.g., negotiation with others.

Strategic decisions—the subject of the theory of games—are those for which the outcome depends on the agent's choice and the non-agentive environment (as in non-strategic decisions), as well as on decisions made by other agents. Encountering someone approaching on the sidewalk, one has to decide whether to swerve left or right. The success of the maneuver will typically depend upon a corresponding decision made by the other agent. The two agents are playing a game—are interacting strategically—in which both are rewarded maximally if each swerves right or if each swerves left; and both are punished, or receive a lesser reward, if one swerves right and the other swerves left.

	C_L (Cooperate)	C_R (Defect)
R_U (Cooperate)	(R,R)*	(S,T)*
R_D (Defect)	(T,S)*	(P,P)#

Fig. 1. Prisoner's Dilemma. T>R>P>S. 2R>T+S. #=Nash equilibrium. * = Pareto-optimal outcome.

To illustrate (with a different game), Figure 1 shows the generic and famous Prisoner's Dilemma game in strategic form. There are are two players, Row and Column. Row has a choice between R_U (mnemonic: up) and R_D (down), while Column chooses between C_L (left) and C_R (right). The outcome (R_D, C_R), in which both players receive P, is said to be a Nash equilibrium because neither player could do better by unilaterally changing its choice of strategy. Row's only alternative choice is R_U, which would yield Row a return of S<P, and similarly for Column. Classical game theory predicts that game outcomes will occur at Nash equilibria. An outcome is said to be (strictly) Pareto-optimal if there is no other outcome at which all (here, both) players can do better. In the Prisoner's Dilemma all of the outcomes *except* the Nash equilibrium are Pareto-optimal. The outcome (R_U, C_L), said to be the result of mutual cooperation, is especially attractive from the players' perspective, since *both* do better than they do at the Nash equilibrium.

[3] So we shall assume for the sake of the example. Nothing is ever so simple. One's loss function—the value one places on staying dry—does matter as much as whether it rains or not.

Why then, in the Prisoner's Dilemma, would the players end up at the Nash equilibrium when they would both be better off at the cooperation outcome? First, it is assumed as part of the setup that the players cannot communicate, or even be aware of who each other is, that no binding agreement on play is possible, and that the players each make their decisions in ignorance of the other's decision. Second, the theory of rationality presumed in game theory posits that players will not select dominated strategies. For Row, R_D is said to be a *dominant strategy* (or dominant choice) because no matter how Column chooses, Row is better off playing R_D, for if Column plays C_L, Row does better with R_D because T>R, and if Column plays C_R, Row does better with R_D because P>S. Similar reasoning applies to Column. Thus, R_U is a dominated strategy for Row and C_L is a dominated strategy for Column. This leaves (R_D, C_R), the Nash equilibrium, as the predicted outcome.

	C_L (Stag)	C_R (Hare)
R_U (Stag)	(R,R)*#	(S,T)
R_D (Hare)	(T,S)	(P,P)#

Fig. 2. Stag Hunt. R>T≥P>S. #=Nash equilibrium. * = Pareto-optimal outcome.

A second game to complete this minimal background may be motivated as follows [Sky01]. Two agents go hunting and take up their places in a blind, which hides them both from each other and from any stags that happen by. Together they can expect to bag a stag, which will feed them each for 3 days. If, however, one of the players reneges and goes hunting for hare, that player can expect to bag two hares, enough to feed him for two days. The other player will receive nothing, neither stag nor hare. If both players renege, each can expect to bag one hare, a day's worth of food. The game, presented in strategic form in Figure 2 is so named in honor of a passage in Rousseau's *A Discourse on Inequality*:

> If it was a matter of hunting a deer, everyone well realized that he must remain faithful to his post; but if a hare happened to pass within reach of one of them, we cannot doubt that he would have gone off in pursuit of it without scruple...

The Stag Hunt is also called the Assurance game. What assurance does a player have that the other player won't renege? The game has been used to model arms races. To see why, relabel. For the row player, change *hunt stag* to *refrain from deploying missile defense* and change *hunt hare* to *fully deploy missile defense.* For the column player change *hunt stag* to *refrain from deploying missile defense penetration system* and change *hunt hare* to *fully deploy missile defense penetration system.*

Hunting stag (or its strategic equivalent) is a cooperative play, as chasing hare is uncooperative. The Stag Hunt game is thus another kind of strategic

repeat forever:

1. Observe the current state, s_t.
2. Select the current action, a_t, from $Q(s, a)$.
3. Take action a_t and obtain reward r_t.
4. Update $Q(s, a)$ based on r_t.

loop

Fig. 3. Pseudo-code for Q-learning in games

context in which issues of cooperation arise. Note that there are two Nash equilibria: (Stag, Stag) (both hunt stag) and (Hare, Hare) (both chase hare), only one of which is Pareto optimal, (Stag, Stag). There is a third, mixed equilibrium, at which both players (independently) hunt stag with probability $x = \frac{P+S}{R+P-T-S}$ and chase hare with probability $1 - x$.

3 Repeated Games

Prisoner's Dilemma and Stag Hunt are each examples of 2×2 games: 2 players, each having 2 strategy choices. This is the simplest kind of game, yet each has generated, and continues to generate, much investigation. Each is problematic in its own way. In the Prisoner's Dilemma there is one Nash equilibrium and it is Pareto inferior to mutual cooperation (C, C). Why can't rational agents figure out a better deal for themselves than mutual defection, (D, D)? In Stag Hunt, there are three Nash equilibria, two of which are Pareto inferior to the third. The prediction of an equilibrium outcome fails to discriminate among the three possibilities.

These kinds of problems are compounded when games are repeated. If we create a repeated (or iterated) game by playing a particular game (called the *stage game*; Stag Hunt or Prisoner's Dilemma are possible examples) over and over again (indefinitely), what happens? At this point, the Folk Theorem (a genuine theorem; it has been proved in various forms) intervenes (see [FM86] or any reasonably advanced and recent textbook on game theory). Roughly, but accurately enough for present purposes, in repeated games of indefinite extent (after each round of play of the stage game there is a finite probability that another round of play will ensue) the number of Nash equilibria explodes; almost every possible combination of payoffs may be the result of some equilibrium. Here, predicting an equilibrium outcome is hardly useful at all. How might rational, or at least intelligent players play and what will happen? To answer these and related questions, it is helpful to investigate the effects of learning.

4 Simple Reinforcement Learning

In *Q-learning,* the objects of learning are state-action pairs, each of which has a value, designated $Q(s, a)$, that is estimated during the learning process. In the context of repeated 2×2 games, the states are constituted by the history of recent play and the possible actions are strategies (rows or columns) in the strategic form representation of the game. Thus, for example, actions in Prisoner's Dilemma would be either Cooperate or Defect, and states are used to describe the recent history of play. In the simplest case, recent history is described merely by what the counter-player did on the last round of play.

The Q-learning algorithm in its barest-bones form may be found in Figure 3. $Q(s, a)$ is initialized in some arbitrary fashion, each player plays an arbitrary strategy, then play begins as described in the figure. On each round of play of the stage game, each agent observes, e.g., what the other agent did on the previous round (1). Given the observed state there will normally be many actions permitted, each of whose current estimated value is recorded in $Q(s, a)$. In the case of 2×2 games, each state will have associated with it two actions. The agent then selects one of the possible actions based on its Q-value (2). Under the ε-greedy selection procedure, the agent picks the available action with the highest Q-value ($Q(s, a)$) with probability $1-\varepsilon$, and the other action with probability ε. (We used the setting $\varepsilon = 0.05$ usually.) Under the SoftMax selection procedure, the agent picks action a in state s with probability

$$\frac{e^{Q_t(s,a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(s,b)/\tau}} \tag{1}$$

where τ is a positive parameter and decreases over time. In our runs we used SoftMax at the start, then ε-greedy as τ reached a minimal threshold. This guaranteed continued exploration by the agents. This is a standard procedure in the Q-learning literature.

After selecting and taking its action (playing a strategy in a round of the game), the agent receives a reward from the round, r, and uses that to update the Q-value of the chosen action, a (item (4) in Figure 3). Our update rule was the standard one:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_b Q(s', b) - Q(s, a)] \tag{2}$$

where α is the learning rate parameter and $Q(s, a)$ on the left is the new, updated value of $Q(s, a)$.

Let us now look at some experimental results. Figure 4 describes Prisoner's Dilemma in a parametric form. We undertook 100 runs of 10,000 rounds of play for each value of δ. The totals are reported in Table 1 for the ε-greedy runs, which were not materially different from the SoftMax runs. We draw the reader's attention in particular to the last column in the table, the meaning which is this. The reward for mutual cooperation is 3 under all

	C_L (Cooperate)	C_R (Defect)
R_U (Cooperate)	(3, 3)*	(0, 3+δ)*
R_D (Defect)	(3+δ, 0)*	(δ, δ)#

Fig. 4. Parametric Prisoner's Dilemma

settings of δ. This represents in an intuitive sense the best each player can do in the long run, assuming the other player avoids exploitation. During the 10,000 rounds of play summarized in the table, an agent getting 3 points per round would extract a point total of 30,000=3×10,000. The last column indicates what percentage of this total was actually obtained by the row player (there was no significant difference for the column player). Quite remarkably, this number is nearly always very high across all values of δ. Although the behavior changed dramatically, from a preponderance of mutual cooperation to a preponderance of mutual defection, the agents were throughout able to obtain an impressive percentage of the effectively available wealth.

Table 1. Summary of results for Prisoner's Dilemma. ε-greedy action selection. Totals for the last 100 rounds of 100 series of 10,000 plays.

CC	CD	DC	DD	δ	Row's % CC
9422	218	183	177	0.05	0.963
9036	399	388	150	0.5	0.963
5691	738	678	2693	1	0.931
3506	179	275	6040	1.25	0.972
1181	184	116	8519	1.5	0.930
2	98	103	9797	1.75	0.805
97	114	91	9698	2	0.735
0	100	92	9808	2.5	0.839
2	96	94	9808	2.95	0.986

The story more or less repeats itself in the very different game of Stag Hunt. Figure 5 presents a parameterized version and Table 2 shows results under the aforementioned conditions (runs of 10,000, etc.). Again, and we think most remarkably, while the behavior of the agents changes with δ, the percentage of the realistic maximal take (here 5 per round) is high and stable throughout (with the possible exception of when δ=3.0).

We may summarize these findings by saying that the Nash equilibrium for the one-shot stage game appears to have little predictive value for play by these agents in the repeated game. Instead, it is the Pareto-superior outcome that can predict the long-run returns the agents will obtain. These findings have been replicated in a wide variety of repeated 2×2 games (see [KL04]

	C_L (Stag)	C_R (Hare)
R_U (Stag)	(5,5)*#	(0,3)
R_D (Hare)	(3,0)	(δ, δ)#

Fig. 5. Parametric Stag Hunt

repeat forever:

1. Select a policy $\pi_i \in \Pi$, where Π is the consideration set of policies.
2. Pick a length of play, l, for policy π_i.
3. Play the next l rounds of the game using π_i.
   Note: At each round, π_i will observe the current state, s_t, take an action a and obtain a reward r_t.
4. Update V^{π_i} based on the individual-round rewards, r_ts, obtained during the l rounds of play of policy π_i.

loop

Fig. 6. Pseudo-code for policy-space learning in games

for detailed elaboration). We now introduce a new variety of reinforcement learning in the context of agents in games.

Table 2. Summary of results for Stag Hunt. ε-greedy action selection. Totals for the last 100 rounds of 100 series of 10,000 plays.

SS	SH	HS	HH	δ	Row's % CC
9390	126	122	362	0	0.978
9546	91	108	255	0.5	0.976
9211	112	125	552	0.75	0.975
8864	119	110	907	1	0.975
8634	115	132	1119	1.25	0.971
7914	122	130	1834	1.5	0.963
7822	122	104	1952	2	0.965
5936	87	101	3876	2.5	0.925
5266	121	106	4507	3	0.736

5 Learning in Policy Space

Figure 6 is the policy space learning analog of state space learning as described in Figure 3. Previously, the objects of learning were state-action pairs, or rather their values, $Q(s, a)$. The objects of learning here are instead *poli-*

cies. A policy $\pi_i(s, a)$ is a mapping from a set of states to an action.[4] We use V^{π_i} to designate the estimated value of policy π_i. Learning in policy space works by getting and improving these estimates. The basic procedure is outlined in Figure 6. Note that step (1) in Figure 6 is analogous to step (2) in Figure 3. Both may use either ε-greedy or SoftMax selection (or a combination), but they do this on different objects, $Q(s, a)$ values in the case of state-space learning and V^{π_i} values in the case of policy-space learning. Updates—step (4) in Figure 6 and step (4) in Figure 3—are also done similarly. For the policy-learning data described below we used this simpler update rule:

$$V^{\pi_i} \leftarrow V^{\pi_i} + \alpha(r/l - V^{\pi_i}) \tag{3}$$

where r is the total reward obtained in the l rounds of play for the strategy. As before α is a "learning rate", typically set between 0.1 and 0.4. The results are not very sensitive to it.

We shall now discuss a number of experiments with learning in policy space. Table 3 presents results from policy-space learning for parameterized Prisoner's Dilemma. In this experiment and in the subsequent experiments reported below, each of the agents/players used a *policy space* consisting of 8 policies, coded `000, 001, 010, ..., 111`. The policies are to be interpreted as follows. Left-most bit indicates what the agent is to play on the first round of play for which the policy applies. In the case of Prisoner's Dilemma, we coded Cooperate with a `1` and Defect with a `0`. The second (middle) bit a policy tells the player what to do if on the previous round of play the counter-player played `0` (or Defect in the case of Prisoner's Dilemma). Finally, the right-most bit of a policy tells the player what to do if on the previous round the counter-player played `1`. Thus, in Prisoner's Dilemma as we coded it, the policy `101`—decimal 5, also known as TIT FOR TAT—directs its holder to cooperate on the first round of play and afterwards to mimic the counter-player's play from the previous round.

Table 3 may be compared to Table 1, although the correspondence is not exact. In Table 3 the average payoff reported is the average over all 800,000 rounds of play, while in Table 1 the average was for the final 100 rounds of a 10,000 round run. Nevertheless, the pattern is clear: policy learners in Prisoner's Dilemma consistently and reliably extract values close to those realized by mutual cooperation. Until δ is very large and the penalty for mutual defection, P, approximates the reward for mutual cooperation, R, the players learn to rely on the TIT FOR TAT strategy, number 5. Here this means players play Cooperate the first time they play the strategy, and after that they play as their counter-player played on the previous round. Once δ is large enough, the players switch to strategy 0, which is to play Defect on every round, no matter what. We note that with δ low the frequency of strategy 5 is a bit lower. This is because the frequency of strategy 7 (Cooperate no

[4] Or more generally, but we do not consider it here, to a probability distribution on a set of actions. See [SB98].

matter what) is comparatively high. In the presence of 5, 7 is a good strategy. The outcome (C, C) of mutual cooperation may be said to be *Pareto-ideal* in the game because (a) it is a Pareto-optimal outcome and (b) the sum of the payoffs is larger than the sum of the payoffs for any other Pareto-optimal outcome. In the case of repeated play of Prisoner's Dilemma, we see that it is generally accurate to predict agents will obtain payoffs on average that are close to the Pareto-ideal outcome for the one-shot game. We will call a hypothesis to this effect a *Pareto-ideal hypothesis.*

Table 3. Summary of results for policy-space learning in Prisoner's Dilemma. Average payoff over 800,000 rounds of play. Modal strategy=most frequently-played strategy; 5 = Tit for Tat, 0 = All Defect.

δ	Average Payoff	Modal Strategy	Freq.	Est. Value	Row's % CC
0.05	2.7224	5	0.5319	2.885	0.907
0.5	2.7577	5	0.7571	2.901	0.919
1.0	2.8108	5	0.8731	2.926	0.937
1.25	2.8139	5	0.8623	2.933	0.938
1.5	2.8083	5	0.8381	2.932	0.936
1.75	2.7950	5	0.8011	2.935	0.932
2.0	2.7314	5	0.6604	2.918	0.910
2.5	2.5324	0	0.8164	2.613	0.844
2.95	2.9524	0	0.8643	3.056	0.984

Table 4. Summary of results for policy-space learning in Stag Hunt. Average payoff over 800,000 rounds of play. Modal strategy=most frequently-played strategy; 5 = Tit for Tat.

δ	Average Payoff	Modal Strategy	Freq.	Est. Value	Row's % CC
0	4.402	5	0.4358	4.721	0.8804
0.05	4.4387	5	0.5104	4.705	0.8877
0.5	4.4747	5	0.6312	4.789	0.8949
1.0	4.5805	5	0.8221	4.854	0.9161
1.25	4.5544	5	0.7467	4.812	0.9109
1.5	4.6057	5	0.8757	4.864	0.9211
2.0	4.6263	5	0.8204	4.869	0.9253
2.5	4.6531	5	0.8822	4.884	0.9306
2.95	4.6733	5	0.8832	4.892	0.9347
3.0	4.7334	5	0.9040	4.901	0.9467

Table 4 presents results from policy-space learning for parameterized Stag Hunt. It may be compared to Table 2. Unlike in the state-space learning case, Table 2, we note a steady improvement in average payoff as δ increases. Concommitantly, there is a nearly monotonic increase in the frequency of using strategy 5, which is TIT FOR TAT and which is in any case the modal strategy used throughout. We note that Stag Hunt is characterized by a strong Pareto-ideal outcome (both hunt stag) and that the Pareto-ideal hypothesis is amply confirmed in these experiments.

Table 7 presents a parametric form of the game called Chicken. This important game has been used to model nuclear war and was made famous in the movie "Rebel without a Cause." The interplay between Nash and Pareto outcomes in the single-shot game presents a new pattern. Three of the four

	C_L (Swerve)	C_R (Straight)
R_U (Swerve)	(2+δ, 2+δ)*	(1,3)*#
R_D (Straight)	(3,1)*#	(0, 0)

Fig. 7. Parametric Chicken

outcomes are Pareto-efficient, and two of these are Nash equilibria. (There is a third Nash equilibrium obtained by mixing the strategies probabilistically. When $\delta = 0$, playing each strategy with probability=0.5 is that Nash equilibrium.) The results from our experiments with learning in policy space for parameterized Chicken may be summarized as follows.

1. $\delta = 0$. One player is dominant, gets an average payoff in the neighborhood of 2.6, and has a modal strategy of 0 (=always go straight). One player is submissive, gets an average payoff in the neighborhood of 1.1, and has a modal strategy of 6 (=straight the first time, after that do the opposite of what the other player did last time). Which player is dominant is entirely random.
2. $\delta = 0.3$. Same general result as $\delta = 0$.
3. $\delta = 0.6$. Chaotic-looking at first, after which both players adopt TIT FOR TAT (strategy 5= swerve the first time, and after that do what the counter-player did the time before). Each player gets an average payoff in the 2.18–2.24 range.
4. $\delta = 0.8$. Multiple reversals of dominance towards the beginning. Eventually both settle down to strategy 5, TIT FOR TAT, each gets an average payoff in the 2.3–2.8 range.
5. $\delta = 0.95$. Both players typically do well, averaging payoffs of 2.56–2.58. Most-used strategies are 3 (= straight the first time, swerve always after that) and 7 (always swerve).

Our last example is Game #47, as identified in [RGG76]. It is a member of a small family of games that, like the Prisoner's Dilemma, have Nash

	C_L	C_R
R_U	(2, 3)#	(4, 1)*
R_D	(1, 2)	(3, 4)*

Fig. 8. Game #47

equilibria disjoint from Pareto-optimal outcomes. The game is of interest as well because unlike the others described here, Game #47 is asymmetric. Will the Nash equilibrium prevail when the game is repeated, or will one of the Pareto-optimal outcomes, and if so, which? In a typical run (typical of all we have observed), Column has an average payoff of about 3.8, Row gets about 2.9. Both players fix on strategy 1 (=play R_D or C_R at first, afterwards play what the counter-player played last round). In short the Pareto-ideal hypothesis is confirmed again.

6 Discussion

Both varieties of reinforcement learning—state-space learning and policy-space learning—performed impressively in a number of ways. First, they reached a limited range of outcomes for the repeated games. For a given value of δ, reinforcement learning is able to support predictions much more specific than are offered by classical game theory and Nash equilibria. Second, the learning agents were remarkably effective at extracting wealth from the games. The Pareto-ideal hypothesis held strongly, except in the case of Chicken, where it held weakly ($\delta \geq 0.6$). Third, the reinforcement learning experiments afford insight into the games explored. Game #47 is a good example. It is especially problematic because of its asymmetry, yet the policy-space learning experiments produce clear and consistent results: the players can find Pareto-optimal outcomes that are superior to the Nash equilibrium and the Column player appears to be in the stronger position, because the average payoff is nearer the (R_D, C_R) outcome than the (R_U, C_R) outcome. Note the subtlety of the issue. Column plays C_R in either case and Row then prefers R_U but is unable to achieve it. What is Column doing to stop Row? Column, in these experiments, is learning which policies pay off best for its purposes and this defeats Row. Fourth, the two learning regimes demonstrate that cognitively limited agents may employ intuitively sensible learning procedures to good effect.[5] This is especially pertinent in light of the fact that much of the reasoning postulated by classical game theory is computationally and cognitively beyond the ken of finite agents.

A few comments on learning in policy space. In the examples given, state-space results are similar to the policy-space results. The latter, however, have a several of interesting distinguishing properties, including the following. The

[5] 'Good' in the sense of effectively extracting value for the player.

number of states in many cases is explosive. The present experiments used a memory of one round of play. Why stop there? A consequence of employing more memory is that the number of states to consider goes up at least at the rate of 2^h where h is the length in rounds of the history under consideration. State-space expansion is a known defeater of learning. Policies, however, offer a way out, which is perhaps not apparent in the examples given here. Policies can act like categories, and may serve to classify an indefinitely large number of individual objects (states in our case). Thus, policies, in virtue of recognizing clusters of states (and failing to discern differences among individual states within a cluster), and learning in policy space may circumvent the "curse of dimensionality" in state space. Moreover, policy space will generally turn out to be (at least pragmatically) much smaller than state space, thereby affording more rapid learning.

Very much remains to be done to explore these ideas and the concept of learning in policy space. What is here is barely a beginning. We think it plausible, however, that is a beginning and an auspicious one at that.

7 Acknowledgements

The work reported here was supported in part by NSF grant number SES-9709548.

References

[Cam03] Colin F. Camerer, *Behavioral game theory: Experiments in strategic interaction*, Russell Sage Foundation and Princeton University Press, New York, NY and Princeton, NJ, 2003.

[FM86] D. Fundenberg and E. Maskin, *The folk theorem with discounting and with incomplete information*, Econometrica **54** (1986), 533–554.

[KL03] Steven O. Kimbrough and Ming Lu, *A note on Q-learning in the Cournot game*, WeB 2003: Proceedings of the Second Workshop in e-Business (Seattle, WA), December 13-14, 2003, Available at http://opim-sun.wharton.upenn.edu/~sok/sokpapers/2004/cournot-rl-note-final.doc.

[KL04] ———, *Simple reinforcement learning agents: Pareto beats Nash in an algorithmic game theory study*, Information Systems and e-Business (forthcoming 2004).

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[RGG76] Anatol Rapoport, Melvin J. Guyer, and David G. Gordon, *The 2×2 game*, The University of Michigan Press, Ann Arbor, MI, 1976.

[SB98] Richar S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, The MIT Press, Cambridge, MA, 1998.

[SC95] T. Sandholm and R. Crites, *Multiagent reinforcement learning in iterated prisoner's dilemma*, Biosystems **37** (1995), 147–166, Special Issue on the Prisoner's Dilemma.

[Sky01] Brian Skyrms, *The stag hunt*, World Wide Web, 2001, Proceedings and Addresses of the American Philosophical Association. http://www.lps.uci.edu/home/fac-staff/faculty/skyrms/StagHunt.pdf. Accessed September 2004.

Learning and Tacit Collusion by Artificial Agents in Cournot Duopoly Games

Steven O. Kimbrough[1], Ming Lu[2], and Frederic Murphy[3]

[1] University of Pennsylvania, Philadelphia, PA, USA,
kimbrough@wharton.upenn.edu
[2] University of Pennsylvania, Philadelphia, PA, USA,
milu@wharton.upenn.edu
[3] Temple University, Philadelphia, PA, USA,
fmurphy@temple.edu

Abstract. We examine learning by artificial agents in repeated play of Cournot duopoly games. Our learning model is simple and cognitively realistic. The model departs from standard reinforcement learning models, as applied to agents in games, in that it credits the agent with a form of conceptual ascent, whereby the agent is able to learn from a consideration set of strategies spanning more than one period of play. The resulting behavior is markedly different from behavior predicted by classical economics for the single-shot (unrepeated) Cournot duopoly game. In repeated play under our learning regime, agents are able to arrive at a tacit form of collusion and set production levels near to those for a monopolist. We note that Cournot duopoly games are reasonable approximations for many real-world arrangements, including hourly spot markets for electricity.

1 Introduction

Strategic behavior by rationally limited agents is interesting for a number of reasons. We draw the reader's attention to two in particular. First, *real* agents are rationally limited. The rationally ideal agents of economics and classical game theory are abstractions, idealizations created for the legitimate purpose of tractable modeling. Without questioning the legitimacy of this, there remains the question of how systems of non-ideal agents will behave. Even if we think that humans in standard market conditions do indeed approximate the rational ideal, no one maintains that birds, bees, monkeys up in trees, and artificial agents do so. And they are interesting, too. Second, in many contexts formal game theory makes predictions that are in various ways unsatisfactory, or unsatisfying, even for rationally ideal agents. ([Col95] and [Kre90] are excellent and accessible reviews of this broadly-accepted assertion.) A case in point occurs when, as is generally the case in repeated games, the number of equilibria in the super-game (the game consisting of repetitions of a sub-game) is large or even infinite. Predicting that the outcome of the super-game will occur at some equilibrium is unsatisfying because it is so

unspecific. Further, classical game theory has little to say about the process of playing and finding good strategies, or how to implement effective agents in strategic contexts, a point developed in [DKL96].

The study of strategic behavior by artificial agents offers a means of addressing many of these and other issues. Methodologically, we may call this a form of *algorithmic game theory,* complementing classical, or *a priori*, game theory, and behavioral game theory (aka: experimental economics) [KL04]. In what follows we present results from this perspective. We examine play and learning by artificial agents in the context of a *repeated* Cournot duopoly game [Cou97]. In §2 we describe the particular version of the model used in our study. §3 discusses briefly the use and appropriateness of the Cournot model in modern electricity markets, in which repetition of play occurs on an hourly basis, which means 8760 plays per year. §§4–5 discuss our learning model for agents in repeated Cournot games. The model is simple and cognitively realistic, yet it departs in an important way from the literature on learning by artificial agents in games. We present our results in §6 and conclude with a discussion in §7.

2 The Duopoly Game: Holt's Cournot Model

The Cournot duopoly model is a staple of classical economics and its textbooks (see, e.g., [Var03], but any standard microeconomics text will do). Because it has been studied both in the laboratory with human subjects (by Holt [Hol85]), and with simple reinforcement learning agents (by Kimbrough and Lu [KL03]), and because it is unproblematically representative of Cournot duopoly models generally, we report on agent learning behavior in the context of a very simple Cournot model, as follows.

There is a duopoly in which the two competing firms are the players and produce a homogeneous product. They individually decide only their individual levels of production, x_1 and x_2. Variable costs are 0 and demand is linear. The total price paid for the joint production, P, is a linear function of the total output:

$$P = A - B(x_1 + x_2) \tag{1}$$

(It is assumed that all variables are greater than 0.) The profit for firm i, $\pi(x_i, x_{-i})$, is a function of both its production, x_i, and the production of the other firm, x_{-i}. (We use conventional notation: $-i$ denotes the player or players in the game *other than* i. This amounts to

$$\pi(x_1, x_2) = x_1[A - B(x_1 + x_2)] \tag{2}$$

for firm 1 in the case, as we have here, of two players. Similarly, the profit for firm 2 is

$$\pi(x_2, x_1) = x_2[A - B(x_1 + x_2)] \tag{3}$$

The Cournot (and Nash) equilibrium is at $x_1 + x_2 = 2A/3B$. The collusive ("monopolist") outcome is $x_1 + x_2 = A/2B$. And the (fully) competitive outcome is $x_1 + x_2 = A/B$. Assuming identical circumstances (so that $x_1 = x_2$), and setting $A = 12$ and $B = 1/2$ (see [Hol85,KL03]), at the Cournot equilibrium each firm produces 8 and realizes a profit of 32. At the collusive outcome, each firm produces 6 and obtains a profit of 36. Finally, at the competitive outcome, each firm produces 12 for a profit of 0.

These are the results predicted by the Cournot model for a one-shot game. What if the game is repeated? In a laboratory experiment, Holt [Hol85] found that human subjects tend to reach production outcomes near the Cournot result, but biased in the direction of the collusive outcome. Kimbrough and Lu [KL03] obtained very similar results for artificial agents in a simple reinforcement learning regime.

3 Background and Application Context: Electricity Markets

Nash/Cournot games have been central to oligopoly theory for decades. The Cournot model assumes that each player i sets its output presuming that the production of the other player, $-i$, is fixed. Equilibrium is defined as any outcome in which player i can do no better given the position of player $-i$, for each player i.[1]

In games with repeated play no learning about the other player is included in the Cournot model. Yet in real life players expend a great deal of effort to understand the other player(s) with whom they compete. In this paper we use agent-based modeling techniques to explore the potential of players to learn about each other and engage each other in a marketplace, occasionally taking the risk of a suboptimal return in any one period. Using simple learning models, we examine the potential of using simple strategies to improve the returns of the players.

We do not claim that a real market will exhibit the behaviors of these players. However, if these simple strategies of the agents outperform the player "optimizations" in the classic Cournot model, then the rationality of the Cournot model needs to be revisited. One would then expect rational business decisions to outperform the agents and the resulting market would then provide higher prices and profits than those predicted by the Cournot model.

Agent-based models of the ilk we describe here will be useful for exploring the role of a futures market in affecting the equilibrium in the spot market for electricity. Allaz and Vila [AV93] show via analytic models that the existence

[1] This assumption is often softened by including notions of conjectural variation, in which player i assumes a rate of response for player $-i$. Player i conjectures a function that gives the decision of player $-i$ as a function of player i's actions. Here the equilibrium is the point at which neither player can improve its position given the other player's assumed response. See [Fri77].

of a futures market leads to higher production and lower prices in the spot market. This is because the players see the influence of their futures decisions on the other player's spot decisions, a kind of conjectural variation.

The literature is mostly confirming of the Allaz and Vila result. Gans, Price, and Woods [GPW98] reproduce the Allaz and Vila results. Le Coq and Orzen [LO02] test the Allaz Vila result in laboratory experiments with students and measure the extent to which futures markets affect spot markets. They find that a futures market leads to increased production, but not to the extent that theory would predict. Adding a futures market is not as effective as increasing the number of players because the students behaved more competitively than theory would predict.

This result and a subsequent stream of literature have had policy impacts. The original design for the electricity market in California restricted futures markets to 20% of expected demand, with the goal of deepening the spot markets. During and after the California electricity crisis, the lack of a futures market was thought to contribute to the market breakdown. Futures markets were added soon after the market stabilized with the belief that they would lead to lower spot prices.

The Allaz Vila model is interesting because, as noted by Harvey and Hogan [HH00], both players are worse off with positive futures positions because they have engaged in a form of the classic Prisoner's Dilemma game, which leads to cooperation in experiments. Other aspects of electricity markets are ripe for study using agent-based modeling. Adjusting the amount of available capacity through maintenance scheduling can be used to raise prices and there is the potential for tacit collusion in setting the schedules. In practice the players bid a set of quantities and prices for every hour while the Cournot model assumes the decision is to set quantities. This leads to complicated games that cannot be analyzed analytically when the fixed costs of starting and shutting down a plant are factored into the decision making.

Bunn and Oliveira [BO03] have used agent-based modeling to explore aspects of electricity restructuring in the United Kingdom. Our work will differ from theirs because we are developing models where agents work in the strategy space (or *policy space*), not just the *state* space for a given decision [KLK04].

In sum, Cournot models (of games) apply quite broadly in oligopoly theory. This itself makes them an interesting target for agent-based modeling. The effect is magnified when Cournot games are *repeated,* as they are hourly in various electricity markets, as well as other markets. With this as amply-motivating context, we turn now to how we modeled agents in Cournot games.

4 Framework for Agent Learning

As we model them, learning players in repeated games have three characterizing attributes:

1. Each player, i, has a *consideration set* of alternative strategies, ${}_i\mathcal{A}$, from which it chooses in making particular plays. Suppressing the pre-subscript, the consideration set $\mathcal{A}$ circumscribes the objects of a player's learning activities; a player learns to play some element of its consideration set. In the present experiments, $\mathcal{A}$ is fixed and given exogenously for each player.
2. Each player, i, has for each element of its consideration set, $j \in {}_i\mathcal{A}$, at each period of the game, t, an *estimate of attractiveness* of the element j. We represent this by ${}_iA_t^j$. After the first period ($t = 0$), each agent has an *attractiveness estimation update rule* by which it may revise its ${}_iA_t^j$s. We use a linear update rule, (BM), after [BM55]. It is standardly employed in the psychology literature and in behavioral studies of games in particular.[2] Related examples of reinforcement learning in games are many.[3] The rule has two parts: (a) If strategy j is *not* played by i at period t, then
$$ {}_iA_{t+1}^j = {}_iA_t^j \tag{4}$$
(b) If strategy j is played by i at period t, then
$$ {}_iA_{t+1}^j = {}_iA_t^j + \alpha(r_{t+1} - {}_iA_t^j + \gamma \max_k {}_iA_t^k) \tag{5}$$
Expression (5) has the form:
NewEstimate = CurrentEstimate + StepSize(reward - CurrentEstimate + DiscountedEstimatedBestPath)
We note that the mean of a series of n observations, $\overline{R}_n = \sum_{t=1}^n r_t/(n)$, can be rewritten equivalently as:
$$ \overline{R}_n = \overline{R}_{n-1} + \frac{1}{n}[r_n - \overline{R}_{n-1}] \tag{6}$$
Instead of the ever-decreasing *StepSize* of $1/n$, expression (5) sets α to be a constant, typically in the neighborhood of 0.25. This has the effect of emphasizing more recent *reward* values, r_{t+1}s, and allows the learner to be responsive to changes in the environment. The *DiscountedEstimatedBestPath* term represents the value of taking action j (i.e., playing strategy $j \in \mathcal{A}$), and thereafter playing the strategy with the best estimated return. The term represents the "shadow of the future" [Axe84]. Below, we report experiments with $\gamma = 0.95$ and $\gamma = 0$.

[2] E.g., [FKP+02], [HW98], [KL04], [MF02], [RC65], [RSB79], and [SC95].
[3] E.g., [BMS00], [CB98], [ER98], [MS04], and [RE95], with reviews in [Cam03,KR95].

3. Each player, i, has an *exploitation-and-exploration policy* which it invokes during play to choose which strategy will govern its behavior at a given time (or round of play).
   Intuitively (and standardly in the machine learning literature[4]) a player's choice of strategy at a particular time should depend on a tradeoff between exploitation and exploration. At an extreme of exploitation, the player will choose the strategy with the highest estimated attractiveness. At an extreme of exploration, the player will choose the least-tried strategy or perhaps choose randomly among the elements of ${}_i\mathcal{A}$. Extreme policies are in general ill-advised. Instead, it has proven effective to use an exploitation-and-exploration policy based on randomization with bias towards choices estimated to be more attractive. Two methods (see [KLM96,SB98]) are in common use. First, under the ϵ–*greedy* policy, the option—in our case the element of ${}_i\mathcal{A}$—with the highest attractiveness is picked with probability $(1-\epsilon)$, and with probability ϵ one of the other elements is chosen randomly and uniformly. (We actually chose from the entire consideration set randomly and uniformly when the ϵ event occurred.) Second, under the softmax policy the probability that an agent choses element j from the consideration set at time t is

$$Prob_t(j) = \frac{e^{\mathcal{A}_t^j/\tau}}{\sum_{j=1}^{n} e^{\mathcal{A}_t^j/\tau}} \tag{7}$$

   where $Prob_t(j)$ is the probability of choosing action j at time (iteration) t. The total number of possible actions is $n = |\ {}_i\mathcal{A}^j|$, the size of the consideration set. $\tau > 0$ is a "temperature" that goes to 0 as t gets large. We use:

$$\tau = T\upsilon^t \tag{8}$$

   where υ is the *annealing factor*, a constant set to nearly 1 (0.9999 in these experiments), T is a scaling factor which we set to 5, and t is the time or iteration number, the present round of play. (In our experiments, once $\tau < 0.0001$ we switched to ϵ–greedy strategy selection, with $\epsilon = 0.1$, for the remainder of the run, as it is throughout the ϵ–greedy policy. Thus, exploration never stopped.)
   Finally, when an agent chooses a strategy to play from the consideration set, the agent also chooses a *commitment length*, c, so that during the next c periods of play, $t+1, \ldots t+c$, the agent commits itself to using the strategy it selected at t. Then, at $t+c$, the agent again invokes its exploitation-and-exploration policy and chooses a new value for c. Commitment lengths were discrete uniform $[a, b]$ variants with $a = 10$ and $b = 20$. We note that the agents' commitment periods were not synchronized with each other.

[4] On reinforcement learning see, e.g., [KLM96] and [SB98].

The principal innovation in the work we report here lies in the constitution of the agents' consideration set, $\mathcal{A}$. Heretofore, learning agents in repeated games and models of learning subjects in repeated games have (implicitly) employed consideration sets consisting of only strategies for the atomic (one-shot) sub-game of the repeated game (or super-game).[5] Previous studies ([SC95] is representative) frame the learning context differently. Strategies in the consideration set have been atomic in the sense that they are not explicitly conditioned on prior play. In the Iterated Prisoner's Dilemma (IPD) case, for example, the agents are able to recognize certain states (aka: "sensations") which record recent history of play. They then learn to associate recognized states with atomic strategies. $\mathcal{A} = \{Cooperate; Defect\}$ in the case of IPD. Thus, recognized states are molecular in the sense of explicitly recognizing more than one period of play, while elements of the consideration set of strategies are atomic in the sense of explicitly representing only one period of play. Our framework inverts this. More precisely, our framework countenances molecular strategies in the consideration set and the experiments we report here are about agents recognizing atomic states but having molecular strategies under consideration. Continuing the IPD example, a *molecular strategy learning framework* would allow the following consideration set: $\mathcal{A} = \{$*If the counter-player cooperated last round, defect this round; If the counter-player cooperated last round, cooperate this round; If the counter-player defected last round, defect this round; If the counter-player defected last round, cooperate this round*$\}$. Our framework would also allow only two states to be recognized, e.g., $\mathcal{S} = \{$*The counter-player cooperated last round; The counter-player defected last round*$\}$. Via the attractiveness values, the ${}_i\mathcal{A}_t^j$s, agents accumulate experience and implicitly remember what has happened. They are *not* in our framework explicitly learning to associate a value with a (state, action) pair. Instead, they learn to associate a value with a strategy and the strategy determines the action, perhaps conditionally. Thus, the agents utilize a kind of *conceptual ascent* in using molecular strategies (conditioned on previous play) instead of atomic strategies, which are statable in terms only of the atomic sub-game.

We turn now to describing the consideration sets our agents used while playing the Cournot game.

5 Molecular Strategies in the Cournot Game

It is convenient here to switch notation just slightly. The strategies are presented from the perspective of one player, x, whose counter-player is y. Thus, e.g., x_t is the production by x at period (play or round) t and y_{t-1} is production by x's counter-player at period $t-1$. We explored play by agents using the following five strategies in the Cournot game, which unlike the usual 2×2

[5] See references cited above. [KL03] might be considered an exception to this statement. If so, it is a borderline one.

Table 1. Average over 100 runs of average profit realized during the last 100 of 10,000 plays

	G-TFT	BestResponse	S-TFT	CopyCat	M-TFT
G-TFT	(36, 36)	(33.138, 28.165)	(36, 36)	(36, 36)	(36, 36)
BestResponse	(28.165, 33.138)	(32, 32)	(32, 32)	(32, 32)	(32.199, 31.899)
S-TFT	(36, 36)	(32, 32)	(36, 36)	(36, 36)	(36, 36)
CopyCat	(36, 36)	(32, 32)	(36, 36)	(23.166, 23.166)	(36, 36)
M-TFT	(36, 36)	(31.899, 32.199)	(36, 36)	(36, 36)	(36, 36)

Table 2. Profit realized by selection policy over the last 1000 of 10,000 plays, averaged over 100 runs; $\gamma = \mathbf{0.95}$; strategies 0–4 available

	Average Profit
Softmax strategy selection	(35.154, 35.109)
$\varepsilon-$greedy strategy selection	(35.495, 35.518)

games, affords agents one-period choices from a continuum. More carefully, the production levels chosen by our agents were represented by floating point numbers.

0. G-TfT. If $y_{t-1} > y_{t-2}$, then $x_t = x_{t-1} + \delta$; else $x_t = x_{t-1} - \delta$
   Comments. Mnemonic: Generous Tit for Tat. Here and throughout these rules, δ is fixed at 0.2. Under this strategy the agent incrementally reduces its production so long as its counter-player has not just increased its production. Note: Here and throughout these rules, production is constrained to be in $[6, 12]$.
1. BestResponse. $x_t = 12 - 0.5y_{t-1}$.
   BestResponse is the strategy hypothesized by Cournot to lead to the Cournot/Nash equilibrium, which occurs at $x_t = y_{t-1} = 8$. In addition, $\pi(x, y)$ is maximized (for x) at $x = 12-0.5y$. There is only one equilibrium and it is stable.
2. S-TfT.
   (a) If $x_{t-1} < y_{t-1}$ and $y_{t-2} \leq y_{t-1}$, then $x_t = x_{t-1} + \delta$.
   (b) If $x_{t-1} > y_{t-1}$ or $x_{t-2} = x_{t-1} = y_{t-1} = y_{t-2}$, then $x_t = x_{t-1} - \delta$.
   (c) Else, $x_t = x_{t-1}$.
   Mnemonic: Suspicious Tit for Tat. A less optimistic and trusting version of G-TfT.
3. CopyCat. $x_t = y_{t-1}$.
   Needs no explanation.
4. M-TfT
   (a) If $x_{t-1} = x_{t-2} = y_{t-1} = y_{t-2}$, then $x_t = x_{t-1} - \delta$.
   (b) If $x_{t-1} = x_{t-2} \neq y_{t-1} = y_{t-2}$, then $x_t = x_{t-1} + 0.5 \cdot (y_{t-1} - x_{t-1})$
   (c) Else:
   Update the step size s_t:
   $s_t = s_{t-1} + \beta \cdot [x_{t-1} - x_{t-2}]/[y_{t-1} - y_{t-2}]$
   Then update x_t:
   $x_t = x_{t-1} + s_t \cdot [y_{t-1} - y_{t-2}]$
   where β is a positive parameter less than 1(we used 0.9).
   Mnemonic: Murphy's Tit for Tat. Another cautious form of Tit for Tat.

Table 3. Profit realized by selection policy over the last 1000 of 100,000 plays, averaged over 100 runs; $\gamma = 0.95$, strategies 0–3 available.

	Average Profit
Softmax action selection	(35.348, 35.323)
$\varepsilon-$greedy action selection	(35.487, 35.507)

Table 4. Profit realized by selection policy over the last 1000 of 100,000 plays, averaged over 100 runs; $\gamma = \mathbf{0}$; strategies 0–4 available

	Average Profit
Softmax strategy selection	(35.243, 35.255)
$\varepsilon-$greedy strategy selection	(35.172, 35.20)

Table 5. Profit realized by selection policy over the last 1000 of 100,000 plays, averaged over 100 runs; $\gamma = \mathbf{0}$; strategies 0–3 available

	Average Profit
Softmax strategy selection	(34.430, 34.392)
$\varepsilon-$greedy strategy selection	(34.689, 34.705)

6 Summary of Results

It is instructive to see how agents will fare pairwise without learning, but sticking to one of the five strategies we consider. Table 1 shows the results reached at the end of 10,000 rounds of play. Specifically. $x_0, y_0 \sim$ *uniform*$(6, 12)$; there were 100 runs of 10,000 rounds of play. The results in Table 1 are the average profits from the 100 last rounds of play, averaged across the 100 runs. The table should be interpreted by anchoring on the outcome $(32, 32)$, predicted by the Cournot model for single-shot cases, and by the Cournot adjustment process, BestResponse, for repeated games. Of the 25 outcome pairs, 15 are at the monopoly (tacit collusion) level of $(36, 36)$, 5 are at the predicted level of $(32, 32)$, 2 are quite nearby, 2 are fairly close, and one is clearly inferior (CopyCat versus CopyCat).

What happens when the consideration set is expanded to all of the five strategies described in §5 and learning is turned on? Remarkably, as summarized in tables 2–5, the agents achieve profits on average that are quite close to the monopoly (collusive) position. They robustly learn to play policies that give them more than 34, often more than 35, points, regardless of which exploitation-and-exploration policy they employ, whether $\gamma = 0.95$ or 0, and whether the run is 10,000 or 100,000 rounds of play. The results in the tables are the average profits from the 100 last rounds of play, averaged across 100 runs of 10,000 or 100,000 rounds.

Plots of agent profitability are also revealing. Figure 1 plots, for a typical run of 10,000 rounds of play, the profits obtained by the two players when the "shadow of the future" is large: $\gamma = 0.95$. Figure 2 plots, for a typical run of 100,000 rounds of play, the profits obtain by the two players when the "shadow of the future" is non-existent: $\gamma = 0$. Notice that play is more volatile when $\gamma = 0$. Both plots, however, exhibit tacit collusion and do so in a way it would be difficult to detect merely from the plots themselves. In both cases, the profits of the agents closely track one another, as we might expect in such a symmetric situation. Further, the results are not materially different when other sets of strategies are available.

7 Discussion

As mentioned above, simple reinforcement learning applied to atomic strategies in the repeated Holt Cournot game results in agents learning to produce slightly less than the Cournot/Nash equilibrium amounts. This results in slightly improved profits for the agents. The consequences are dramatic from making a conceptual ascent by learning molecular strategies, strategies that are themselves explicitly conditioned on outcomes of previous games. Our agents learned to produce, on average, amounts very near (an equal split of) the monopolist's quantities, effectively maximizing their profits jointly and individually. We emphasize that the fundamental learning regime was identical in the two cases. What differs is the *objects* of learning, what is in the consideration set $\mathcal{A}$, rather than the process of learning itself. In both cases it was simple, classical reinforcement learning. One might think of our agents as being wiser—in considering more options—rather than being smarter. Our agents engage in *policy space learning* rather than the conventional *state space learning* of reinforcement learning [KLK04].

Of course, much remains to be done by way of exploring conceptual ascent to molecular strategies. To that end we close with two comments. First, our findings are robust. In unpublished work we have examined a broad range of 2×2 games and obtained results in conformance with those reported here. As stated in the title of [KL04], in simple reinforcement learning in 2×2 games, "Pareto beats Nash." That is, when Pareto-optimal outcomes conflict with Nash equilibrium outcomes (e.g., as in Prisoner's Dilemma), these simple learning agents will under many conditions arrive at outcomes much closer to the Pareto outcomes than to the Nash outcomes. When the Pareto outcomes coincide with some of the Nash outcomes (e.g., as in Stag Hunt), the players tend to find those Nash outcomes that are also Pareto. These effects are magnified greatly with conceptual ascent to molecular strategies in the consideration set. In the Cournot game, the Cournot outcome is the Nash equilibrium, and the collusive outcome—each agent produces 6—is the Pareto-optimal outcome.

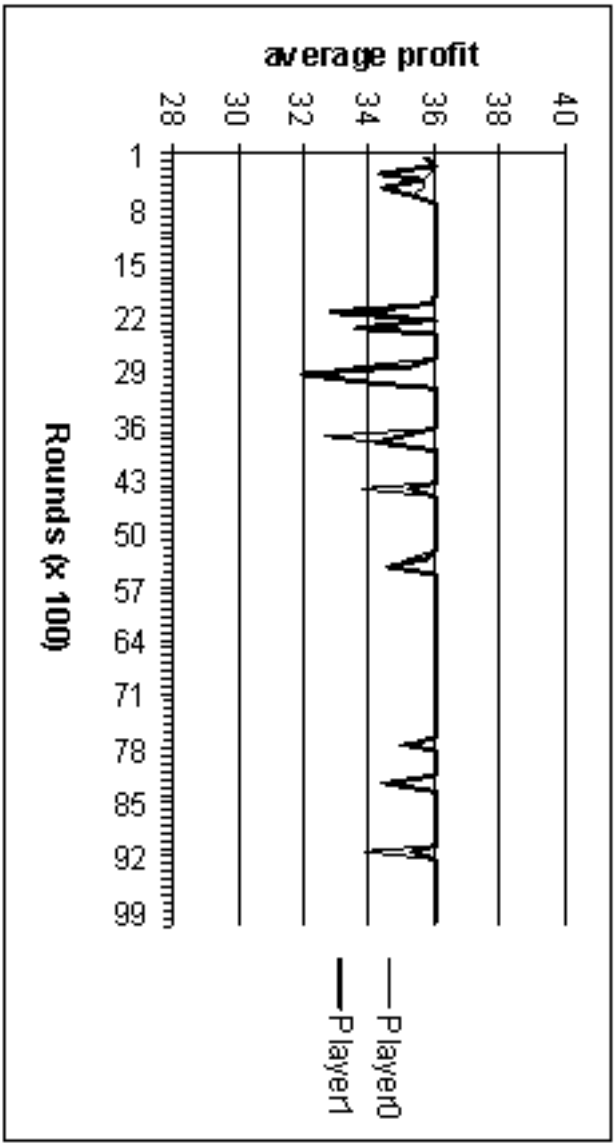


Fig. 1. Average profit realized by player over 10,000 repetitions in a typical run, $\gamma = \mathbf{0.95}$; strategies 0–4 available; ε–greedy strategy selection.

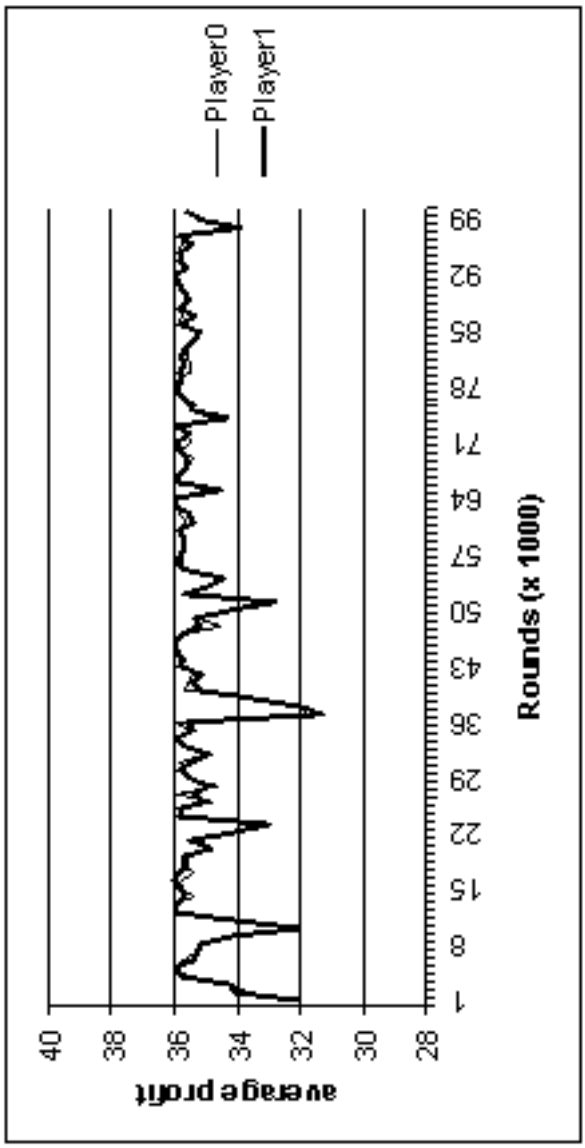


Fig. 2. Average profit realized by player over 100,000 repetitions in a typical run, $\gamma = \mathbf{0}$; strategies 0–4 available; ε–greedy strategy selection

Second, the Cournot game is—as is Prisoner's Dilemma—an example of a social dilemma [Daw80], "in which decisions that make sense to each individual can aggregate into outcomes in which everyone suffers".[6] The literature in economics and game theory has tended to assume, or postulate, that players will reach the Cournot/Nash equilibrium because they will employ the BestResponse strategy. Our agents have ample access to that strategy and there are times when they approach Cournot outcomes. Yet on balance they overwhelmingly are able to rise above their social dilemmas and find ways to tacitly collude, for mutual profit. If our simple agents can do this in a repeated Cournot game, is it not plausible that firms playing such a game hourly for large amounts of money, as in the spot market for electricity, will find similar ways to collude tacitly at the expense of their customers? The electricity market is hardly unique in this regard. It is time to use computational agent technology to revisit some fundamental beliefs.

8 Acknowledgements

The work reported here was supported in part by NSF grant number SES-9709548.

References

[AV93] B. Allaz and J.-L Vila, *Cournot competition, forward markets and efficiency*, Journal of Economic Theory **59** (1993), 1–16.

[Axe84] Robert Axelrod, *The evolution of cooperation*, Basic Books, Inc., New York, NY, 1984.

[BM55] R. R. Bush and F. Mosteller, *Stochastic models for learning*, Wiley, New York, NY, 1955.

[BMS00] B. Banerjee, R. Mukherjee, and S. Sen, *Learning mutual trust*, Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies, 2000, citeseer.nj.nec.com/banerjee00learning.html, pp. 9–14.

[BO03] D. W. Bunn and F. Oliveira, *Evaluating individual market power in electricity markets via agent-based simulation*, Annals of Operations Research **121** (2003), 57–78.

[Cam03] Colin F. Camerer, *Behavioral game theory: Experiments in strategic interaction*, Russell Sage Foundation and Princeton University Press, New York, NY and Princeton, NJ, 2003.

[CB98] Caroline Claus and Craig Boutilier, *The dynamics of reinforcement learning in cooperative multiagent systems*, Proceedings of the Fifteenth National Conference on Artificial Intelligence (Menlo Park, CA), AAAI Press/MIT Press, 1998, pp. 746–752.

[Col95] Andrew M. Colman, *Game theory and its applications in the social and biological sciences*, second ed., Routledge, London, UK, 1995.

[6] [MF02]

[Cou97] A. Cournot, *Researches into the mathematical principles of the theory of wealth*, Macmillan, New York, NY, 1897, English edition edited by N. Bacon. Originally published in French as *Recherches sur Principes Mathématiques de la Théorie des Richesses* in 1838.

[Daw80] Robyn M. Dawes, *Social dilemmas*, Annual Review of Psychology **31** (1980), 169–193.

[DKL96] Garett O. Dworman, Steven O. Kimbrough, and James D. Laing, *Bargaining by artificial agents in two coalition games: A study in genetic programming for electronic commerce*, Genetic Programming 1996: Proceedings of the First Annual Genetic Programming Conference, July 28-31, 1996, Stanford University (John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, eds.), The MIT Press, 1996, pp. 54–62.

[ER98] Ido Erev and Alvin E. Roth, *Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria*, The American Economic Review **88** (1998), no. 4, 848–881.

[FKP+02] Christina Fang, Steven O. Kimbrough, Stefano Pace, Annapurna Valluri, and Zhiqiang Zheng, *On adaptive emergence of trust behavior in the game of stag hunt*, Group Decision and Negotiation **11** (November 2002), no. 6, 449–467.

[Fri77] J. W. Friedman, *Oligopoly and the theory of games*, North Holland (now Elsevier), 1977.

[GPW98] J. S. Gans, D. Price, and K. Woods, *Contracts and electricity pool prices*, Australian Journal of Management **23** (1998), no. 1, 83–96.

[HH00] S. M. Harvey and W. W. Hogan, *California electricity prices and forward market hedging*, Technical report: working paper series, Center for Business and Government, John F. Kennedy School of Government, Harvard University, Cambridge, Massachusetts 02138, October 2000.

[Hol85] Charles A. Holt, *An experimental test of the consistent-conjectures hypothesis*, The American Economic Review **75** (1985), no. 3, 314–325.

[HW98] J. Hu and M. P. Wellman, *Multiagent reinforcement learning: Theoretical framework and an algorithm*, Fifteenth International Conference on Machine Learning, July 1998, pp. 242–250.

[KL03] Steven O. Kimbrough and Ming Lu, *A note on Q-learning in the Cournot game*, WeB 2003: Proceedings of the Second Workshop in e-Business (Seattle, WA), December 13-14, 2003, Available at http://opim-sun.wharton.upenn.edu/~sok/sokpapers/2004/cournot-rl-note-final.doc.

[KL04] ———, *Simple reinforcement learning agents: Pareto beats Nash in an algorithmic game theory study*, Information Systems and e-Business (forthcoming 2004).

[KLK04] Steven O. Kimbrough, Ming Lu, and Ann Kuo, *A note on strategic learning in policy space*, Formal Modelling in Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, 2004.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KR95] John H. Kagel and Alvin E. Roth (eds.), *The handbook of experimental economics*, Princeton University Press, Princeton, NJ, 1995.

[Kre90] David M. Kreps, *Game theory and economic modeling*, Clarendon Press, Oxford, England, 1990.

[LO02] C. Le Coq and Henrik Orzen, *Do forward markets enhance competition? experimental evidence*, Technical report: working paper series, The Economic Research Institute, Stockholm School of Economics, SSE/EFI Working Paper, Department of Economics, Sveavagen, P.O. Box 6501, 113 83 Stockholm, Sweden, August 2002.

[MF02] Michael W. Macy and Andreas Flache, *Learning dynamics in social dilemmas*, Proceedings of the National Academy of Science (PNAS) **99** (2002), no. suppl. 3, 7229–7236.

[MS04] Rajatish Mukherjee and Sandip Sen, *Towards a pareto-optimal solution in general-sum games*, 2004, citeseer.nj.nec.com/591017.html.

[RC65] Anatol Rapoport and Albert M. Chammah, *Prisoner's dilemma: A study in conflict and cooperation*, The University of Michigan Press, Ann Arbor, MI, 1965.

[RE95] Alvin E. Roth and Ido Erev, *Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term*, Games and Economic Behavior **8** (1995), 164–212.

[RSB79] Amnon Rapoport, William E. Stein, and Graham J. Burkheimer, *Response models for detection of change*, D. Reidel Publishing Company, Dordrecht, Holland, 1979.

[SB98] Richar S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, The MIT Press, Cambridge, MA, 1998.

[SC95] T. Sandholm and R. Crites, *Multiagent reinforcement learning in iterated prisoner's dilemma*, Biosystems **37** (1995), 147–166, Special Issue on the Prisoner's Dilemma.

[Var03] Hal R. Varian, *Intermediate microeconomics: A modern approach*, W. W. Norton & Company, New York, NY, 2003.

A Note on Working Memory in Agent Learning

Fang Zhong

Georgia Institute of Technology, Atlanta, GA, USA,
fang.zhong@mgt.gatech.edu

Abstract. An important dimension of system and mechanism design, working memory, has been paid insufficient attention by scholars. Existing literature reports mixed findings on the effects of the amount of working memory on system efficiency. In this note, we investigate this relationship with a computational approach. We design an intelligent agent system in which three agents, one buyer and two bidders, play an Exchange Game repeatedly. The buyer agent decides whether to list a request for proposal, while the bidders bid for it independently. Only one bidder can win on a given round of play. Once the winning bidder is chosen by the buyer, a transaction takes place. The two parties of the trade can either cooperate or defect at this point. The decisions are made simultaneously and the payoffs essentially follow the Prisoner's Dilemma game. We find that the relationship between working memory and the efficiency of the system has an inverted U-shape, i.e., there seems to be an optimal memory size. When we mixed agents with different memory sizes together, agents with the same amount of working memory generate the most efficient outcome in terms of total payoffs.

1 Introduction

One of the enduring challenges in electronic commerce is modeling and implementing formal conversations among agents [KL97]. In facing this challenge, researchers need to deal with the design of intelligent agents who can not only communicate with each other effectively, but also respond promptly, efficiently, and intelligently in dynamic environments. Previous research has shown the potential of intelligent agents with learning abilities for achieving cooperation or a high level of trust in trading environments without legal enforcement.[1] The focus of these studies is on applying Q-learning (a form of reinforcement learning[2]) to the design of agents. Learning, however, is only one aspect of intelligence. Another important dimension is the working memory of an agent, i.e., how much historical information is incorporated by an agent in its decision-making processes.

This design question is not trivial. Sandholm and Crites[3] conducted a study in which two artificial agents play the Iterated Prisoner's Dilemma

[1] [KWZ02,SC95,ZKW02]
[2] [KLM96,SB98,WD92]
[3] [SC95]

(IPD) game. The two agents could be either Q-learners or TIT-FOR-TAT strategy players. They showed that agents with larger memories seemed to be able to gain larger total payoffs when playing against agents with smaller memories. However, the focus of their paper is not on the influence of working memory, but on applying Q-learning in multi-agent system design. One major difficulty of expanding memory size in the context of reinforcement learning is the resulting explosive state space [KLK04]. Due to the inherent computational complexity of large-scale investigation in this topic, we first need to find evidence in and results for a relatively small system. That is the purpose of this note.

In our computational approach, learning agents with different memory sizes play a multi-stage trading game repeatedly. The last stage of the game is essentially a simultaneous Prisoner's Dilemma game. Our results show that the impact of memory size on agent performance has an inverted U-shape. There seems to be an optimal memory size. Specifically, it appears that for a two-by-two game, conditioning the bidder's decision on information from the last two periods is optimal compared to memory sizes of one or three periods. Moreover, when agents with different memory sizes trade with each other, having more memory does not pay off. These findings contravene the results reported in Sandholm and Crites's paper [SC95].

Another contribution of this note is the provision of a model in which agents have the option of choosing their trading partners. In social exchanges involving trust, participants not only decide how to bargain with one another; they also choose with whom to bargain. Most studies focus on the bargaining or exchange process itself, with participants either fixed or matched randomly against one another.[4] However, the situation in which a potential participant in an exchange can choose a partner, or even whether to enter the exchange at all, has rarely been investigated. It is important to understand, in this more realistic context, whether intelligent agents can make the right choices when facing different potential partners.

The rest of this note is organized as follows. §2 provides motivation for and description of the Exchange Game we study throughout the note. §3 describes briefly the design of the agents. The experimental design and results are discussed in §4, after which we conclude the note.

2 The Exchange Game

The model proposed here is motivated by the type of exchange existing in online markets, in which a buyer initiates an auction by posting a Request for Proposal (RFP). Listing an RFP is optional. The buyer incurs this cost only if it decides to solicit proposals. This cost comes from preparing the RFP and posting it in the right market. Bidders subsequently submit bids. To consider the endogenous entry decision of bidders, we include bidding cost

[4] E.g., [CB98], [HW98], [SC95], and [ZKW02].

in our model, i.e., if a bidder decides to bid, it incurs a fixed bidding cost. Upon receiving the bids, the buyer chooses one winning bidder and both parties proceed to a transaction. In their empirical study, Sandholm and Crites [SC95] conducted an empirical study of reinforcement learning (RL) in the Iterated Prisoner's Dilemma (IPD) game. Their multi-agent system consisted of two agents, which could be either Q-learners or Tit-for-Tat strategy players. They found that:

1. Agents were able to learn to cooperate,
2. Agents with larger memories seemed to be able to gain larger total payoffs when playing against agents with smaller memories,
3. More cooperative results were observed when the memory size is 1 for both agents,
4. Longer exploration is better at inducing cooperation, and
5. Agents using lookup tables outperformed agents using a recurrent neural network.

The exchange regime we have modelled is a multi-stage game. There have been arguments in the literature about the aptness of the mixed game model as an abstraction of real-world games [BS00]. The existing two-stage game models usually consist of strategic choices, such as a capacity decision, that influence subsequent games and in particular the pricing strategy in the second stage of the game [Sha89,Sut91]. The multi-stage trading game proposed here is different from these previous game models. First, we forego the details of the decisions in the early stage. Instead, the strategic choices in our model concern game entry: to list an offer or not, to bid or not, and which partner to select. Such simplification enables us to keep our focus on the emergence of cooperative behavior rather than the pricing or capacity strategies. Second, we study the model in the context of repeated games. Consequently, not only do early stages influence subsequent games, the later stages also have impact on the early stage decision-making through the repetitions.

In the basic setting there is a community of N sellers or service providers and one buyer. Each seller can potentially fulfill the requirements of the buyer. At the beginning of each period, t, the buyer decides whether to list an RFP for bid. If the buyer lists an RFP, it incurs a fixed cost of C_s. After observing the RFP, l ($l \leq$M) sellers send in their bids (one bid per bidder who bids). Each bid incurs a fixed cost C_b. Both C_s and C_b are private information. Upon acknowledgement of bidding information, the buyer chooses which bid to accept or chooses nothing. If a bid is accepted, the two parties proceed to the trading stage in order to arrange a transaction. In this stage, each party can choose to defect or cooperate. Without loss of generality, the trading game is essentially a Prisoner's Dilemma game defined in Table 1. The payoffs are unknown to the agents at the beginning of the game. They can only learn the values of the payoffs through repeatedly playing the Exchange Game.

The net profits of both parties will be their payoffs from the trading stage after subtracting the costs they incur. Obviously the cost is crucial to the

Table 1. Payoffs of the trading game. $S < P < B < T$ and $2B > T + S$

		Bidding Agent	
		C	D
Seller Agent	C	(B, B)	(S, T)
	D	(T, S)	(P, P)

net profits of the agents. To simplify, we set C_s=C_b=C. In the experiments we discuss below, both buyer cost and bidder cost can take four values corresponding to the payoff structure: $C = 0$, $C = S$, $C = P$, and $P < C < B$.

Based on this model, the buyer agents have to decide:

1. Whether to announce an RFP or not.
2. Which trading partner to select.
3. Whether to defect or not if there is a trade.

Similarly, bidders decide:

1. While facing an RFP, whether to bid or not.
2. Whether to defect or not if there is a trade agreed upon.

Accordingly, three stages of the game are defined: the offering stage (only for the buyer agents), the bidding stage, and the trading stage. The three stages constitute a mixed game of both sequential and simultaneous games as illustrated in Figure 1.

To simplify, we investigated a 3-agent Repeated Exchange Game. The three agents include one buyer agent S and two bidder agents B_1 and B_2. We start with a basic model, for which in each time period, t, the buyer agent lists one RFP without incurring any expense. The order of play of the stage game is as follows:

1. Facing the RFP, agents B_1 and B_2 individually decide whether to bid or not. An agent that bids incurs a cost of C_b.
2. Agent S chooses one agent with which to trade. If S chooses a non-bidding agent, the game is over. Agent S can also choose not to trade with any of the other agents.
3. If agent S accepts a bid, a trade is arranged for the two parties. Both parties independently decide whether to defect or not, and the payoffs of the trade are determined by Table 1.
4. If a stopping condition is not met, go to step 1.

In the extended model, the buyer agent is obliged to pay a fixed cost when listing an offer. Thus the stage game will be slightly changed to:

1. If the buyer decides to solicit a request, it incurs cost C_s, otherwise, the game stops.

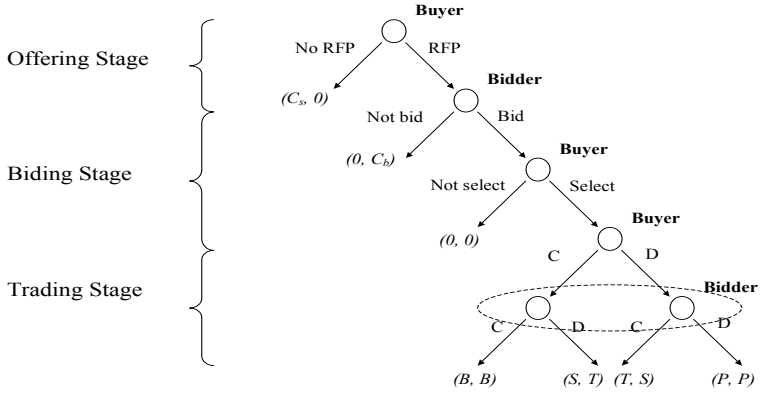


Fig. 1. Extensive form of the stage game

2. Facing the RFP, agents B_1 and B_2 individually decide whether to bid or not. An agent that bids incurs a cost of C_b.
3. Agent S chooses one agent with which to trade. If S chooses a non-bidding agent, the game is over. Agent S can also choose not to trade with any of the other agents.
4. If agent S accepts a bid, a trade is arranged for the two parties. Both parties independently decide whether to defect or not, and the payoffs of the trade are determined by Table 1.
5. If a stopping condition is not met, go to step 1.

In the next section we briefly discuss the design of our agents.

3 Learning Mechanism

We designed a system in which three Q-learning agents learn about their environment, i.e., about the behavior of other agents and the payoffs of the game. Q-learning is a category of reinforcement learning, which is a form of machine learning. It is a mechanism in which actions that have led to favorable results are more likely to be chosen in the future. One of the most important developments in reinforcement learning was the discovery (or invention) of

Q-learning [WD92], in which the learned state-action-value function *Q(s, a),* or *Q-value*, directly approximates the optimal state-action-value function. In a stationary environment, the learned Q-value is guaranteed to converge to the optimal as long as all state-action pairs continue to be updated [SB98]. However, when agents are learning simultaneously, the environment becomes non-stationary and thus convergence is not guaranteed [SC95]. In this study, we report statistical results of multiple runs of the same experiment in order to overcome any concerns in this regard. The rest of this section describe the learning algorithm in each of the three stages of our Exchange Game.

3.1 Trading Stage

When at the trading stage, agents have memory of the last move, which leads to four states $s \in \{CC, CD, DC, DD\}$. For each state, two actions are available $a \in \{C, D\}$. After being initialized to arbitrary numbers, Q-values are estimated on the basis of the following algorithm:

1. From the current state s, select an action. This will cause an immediate payoff r, and arrival at a next state s'.
2. Update Q(s, a) based on:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r + \gamma \max_b Q_t(s',b) - Q_t(s,a)] \tag{1}$$

3. Go to 1 until the last iteration of the repeated game.

3.2 Offering Stage and Bidding Stage

In each of these two stages, agents need to learn if proceeding to the trade stage is profitable. For the buyer agent, it also needs to learn for each bidder how profitable it is to trade. To model the decision making process, we let agents learn the expected average payoff they get from trading. For example, the buyer agent learns the expected average payoffs from trading with either bidder agent separately. Agents decide to enter the exchange if and only if the learned average payoff is larger than the cost they need to pay for participation. If trading with both bidder agents is profitable, the one which yields higher average payoff for the buyer is more likely be chosen.

If the buyer agent chooses to list an offer and one bidder agent chooses to bid, yet the buyer agent doesn't accept the bid, there will be no trade. In this case, neither agent can gain anything from the exchange. Instead, they incur costs and end up with negative rewards.

Since agents have only one state at these two stages, we can drop the state variable in an action function $Q(s, a)$ and use $Q(a)$ instead. Let α denote the learning rate and r the reward of taking an action, the Q-value at period t for choosing action a_i at these two stages is updated by:

$$Q_t(a_i) = Q_{t-1}(a_i) + \alpha[r - Q_{t-1}(a_i)] \tag{2}$$

3.3 Exploration

At all stages, the selection of an action is based on the Boltzmann distribution, in which the probability of selecting action a_i in state s is

$$p(a_i) = \frac{e^{Q(s,a_i)/\tau}}{\sum_a e^{Q(s,a)/\tau}} \tag{3}$$

τ is a computational 'temperature' to control the degree of exploration. Specifically, in the offering stage, τ is a function of the number of the iterations played heretofore; in the bidding stage, τ is a function of the number of iterations that have valid offers; while in the trading stage, τ is a function of the number of trades completed between two agents. Although the starting value of τ is the same for all stages, the annealing schedule is different. It is smaller in the later stages, indicating shorter exploration. The rationale for such a design is that later stage subgames should stop exploration first so that the earlier stages can use the relatively stable Q-values of the later stages to keep exploring and learning. When τ_i reaches 0.01, exploration stops and instead the best action is selected for the remainder of the run. This method is usually referred to as Softmax action selection [SB98].

In the following sections, we discuss details of the experimental design and the results. Unless otherwise stated, the experiments were run with fixed values of the parameters. The selection of the values is benchmarked with those in [SC95]. Actual values can be found in the Appendix. Also, we ran each experiment 100 times with 400,000 iterations in each run and recorded the last 100 iterations of each run. The frequencies of different game plays—in which the first strategy belongs to the seller agent and the second one belongs to bidder agent—from these 10,000 iterations are reported.

4 Experiments on Working Memory

As mentioned earlier, the purpose of these experiments is to investigate the effects of memory size. We want to see whether having fixed working memory—specifically the minimum memory—is efficient enough, as predicted by an economic model [Del03], or whether storing the entire dynamic history information is better. Could the relationship between memory size and agent performance take a different form than these two alternatives? To answer this question, we design agents with a memory size of up to three, i.e., agents can remember the trading outcome of up to the last three periods. The reason we chose three levels of memory size is because the action space in the trading stage is fairly small: cooperate or defect. The basic setting includes three Q-learning agents, one buyer agent and two bidder agents, playing the Exchange Game repeatedly. We compare the final game play outcome and the total surplus achieved.

We conducted three experiments. The purpose of the first is to serve as a benchmark for subsequent experiments. Then, we looked at cases in which agents have same memory size. Finally, we looked at cases in which agents have different memory sizes. To evaluate the impact of memory size on efficiency, we compare the final game play outcome and the total surplus achieved by agents in each experiment.

4.1 Benchmark Setting

In the benchmark setting, agents have no memory, i.e., memory sizes of the three agents are zero, and no costs were involved. The results of this experiment are reported in Tables 2, 3, and 4 in conjunction with results from the next two experiments.

4.2 Basic Setting: Identical Agents

In this set of experiments, we are going to investigate the following three cases:

1. All three agents have memory size of 1 (case 1-1-1)
2. All three agents have memory size of 2 (case 2-2-2)
3. All three agents have memory size of 3 (case 3-3-3)

The total payoff of each agent under various different cost schemes is reported in Tables 2, 3, and 4. All the tables have the same structure. Columns 2 to 5 correspond to the situation in which only bidder agents incur costs. Columns 6 to 8 correspond to the situation in which both the buyer and the bidder agents incur costs. Rows 3 to 5 indicate the total points of the three agents respectively in various cost settings.

1. All three agents have memory size of 1 (case 1-1-1)

Table 2. Results of case 1-1-1

	Bidding cost only				Bidding and listing costs		
	C=0	C=S	C=P	P<C<B	C=S	C=P	P<C<B
Buyer agent	3278	3382	3404	961	2308	1076	-1
Bidder 1	1694	1052	872	50	1068	656	1
Bidder 2	1564	1298	571	-53	1233	720	1

2. All three agents have memory size of 2 (case 2-2-2)

Table 3. Results of case 2-2-2

	Bidding cost only				Bidding and listing costs		
	C=0	C=S	C=P	P<C<B	C=S	C=P	P<C<B
Buyer agent	3696	3614	3681	1820	2507	1304	7
Bidder 1	1957	1181	830	183	1178	619	2
Bidder 2	1760	1436	772	121	1457	881	0

3. All three agents have memory size of 3 (case 3-3-3)

Table 4. Results of case 3-3-3

	Bidding cost only				Bidding and listing costs		
	C=0	C=S	C=P	P<C<B	C=S	C=P	P<C<B
Buyer agent	3231	3142	2908	435	1989	17	-8
Bidder 1	1578	995	449	71	1046	26	1
Bidder 2	1563	1064	520	23	978	27	1

By comparing the data in the three tables as well as those in the case of memory size of zero, we can see that agents achieved the most total surplus in case 2-2-2, in which every agent has memory for the previous 2 rounds of play. The finding is consistent across all cost levels. This phenomenon can be observed from Figure 2 directly, which demonstrates that all three agents obtained their highest total points when they have memory size of 2. The figure suggests that the relationship between working memory and agent performance has an inverted U-shape.

To explain the existence of such phenomena, we looked at the game plays in the last 100 iterations of 100 runs of each case. The percentages of the four possible game plays, CC, CD, DC, and DD, in the 10,000 iterations in different memory size settings are shown in Figure 3. When memory size is 0, the agents converged to the Nash equilibrium (NE) in most runs. When the memory size is increased to 1, agents start to deviate from the NE outcome and achieve more Parato frontier game play, which increases the total points that each agent earns. When the memory size is 2, the deviation becomes more significant and leads to even higher returns for the agents. However, when the memory size is 3, the percentage of NE game play bounces back. Combined with a decrease of mutual cooperation, it results in a lowered return for each agent.

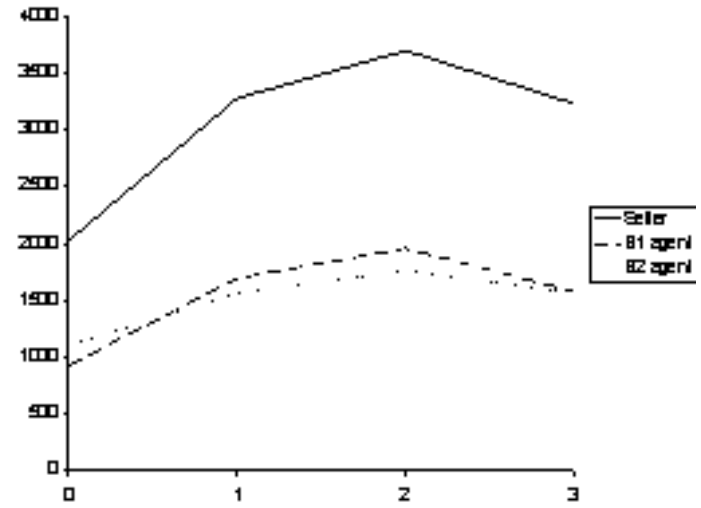


Fig. 2. Total points of each agent when memory size is 0, 1, 2, and 3 respectively

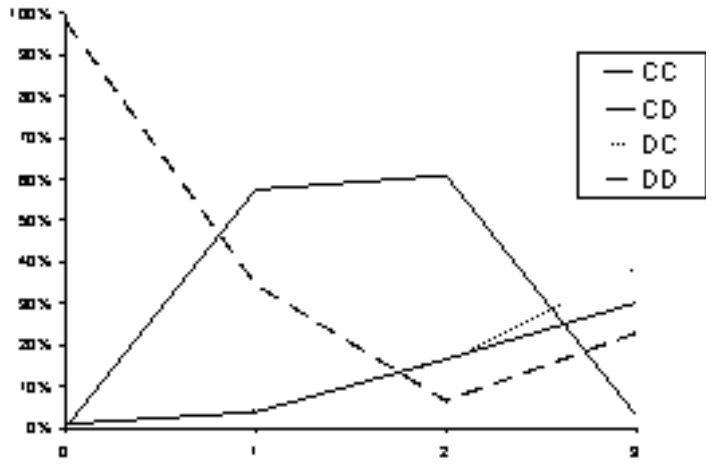


Fig. 3. Percentage of four possible game plays in the last 100 iterations of 100 runs when memory size is 0, 1, 2, and 3 respectively. (No costs were involved.)

4.3 Extended Setting: Heterogeneous Agents

In the next group of experiments, we looked at cases in which agents have different memory sizes. Each case is identified by a series of three numbers indicating the assignment of memory sizes to the three agents. The first number is the memory size of the buyer agent, the second number is for the B_1 agent and the last number is for the B_2 agent. For example, case 1-2-1 means that the seller agent has memory size of one; the B_1 agent has memory size of two; and the B_2 agent has memory size of one again. Eight cases were examined: 1-1-2; 1-2-1; 1-1-2; 2-1-1; 2-2-3; 2-3-2; 2-3-3; and 3-2-2. The eight cases were further categorized into four groups. The allocation of the groups is described in Table 5.

Table 5. Grouping of the experiments with asymmetric assignment of memory size

Group number	Case number
1	1–1–2 and 1–2–1
2	2–3–2 and 2–2–3
3	2–1–1 and 1–2–2
4	2–3–3 and 3–2–2

Based on the findings from agents with identical memory size, the cost scheme does not influence the role of memory size. These findings are consistent across all cost schemes. In this set of experiments, we take away both the bidding and listing costs. The total points of each agent in different cases are summarized in Table 6.

Table 6. Total points of the three agents in the eight cases of asymmetric memory size assignment

	Group 1		Group 2		Group 3		Group 4	
	1-1-2	1-2-1	2-2-3	2-3-2	2-1-1	1-2-2	2-3-3	3-2-2
Buyer agent	3348	3235	3562	3623	2658	2588	3207	3404
Bidder 1	3015	299	3250	373	1271	1349	1651	1679
Bidder 2	363	2893	337	3230	1277	1406	1620	1522

A more visual presentation of the results is illustrated in Figure 4. Data from groups 1 and 2 are similar in the sense that the buyer agent has the option of choosing between bidder agents with different memory sizes. In these cases, the buyer agent chooses the one with the same memory size as that of itself. However, because agents in group 2 have larger memory sizes than those in group 1, they achieved more total surplus. Groups 3 and 4 are

identical in the sense that the two bidder agents have the same memory sizes. It does not make much difference for the buyer agent to choose one over the other because now it is impossible for the buyer agent to pair with an agent with the same memory size, i.e., the memory sizes are nonidentical between the two agents who trade with each other eventually. In these cases, agents in groups 3 and 4 achieved less total surplus than the agents in groups 1 and 2 respectively where pairing with agents with identical memory size is possible. What seems consistent throughout is that agents in group 4 obtained more total points than the agents in group 3 because they have larger memory sizes. To summarize, it seems that agents with the same memory size tend to pair together and achieve higher return. When this kind of pairing is impossible, agents with larger memory sizes would pair together to get more return.

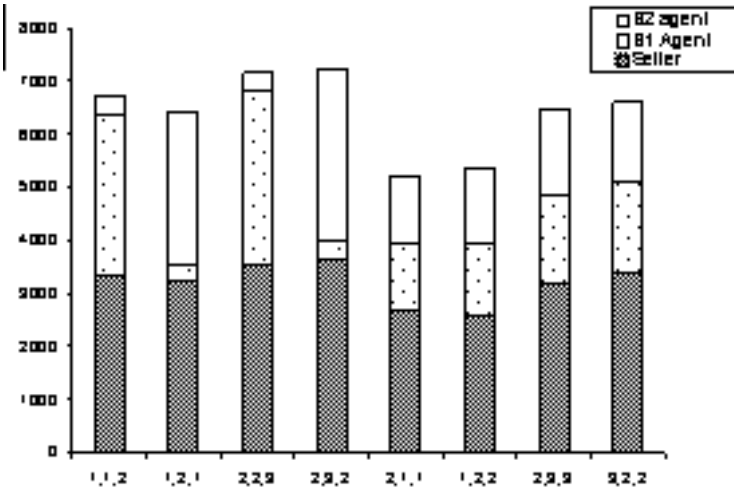


Fig. 4. Total points of the three agents in the eight cases of different memory size assignment, alternative presentation

5 Discussion

Through the above experiments, we have essentially found two very interesting phenomena. First, the impact of working memory on system efficiency is not as simple as either economic models or empirical studies predicted. If we interpret memory size as a dimension of intelligence, the inverted U-shaped curve we observed implies that neither too little nor too much intelligence is efficient. With minimum working memory, the historical information is not sufficient for agents to condition their decisions upon. Research on the classic iterated Prisoner's Dilemma (IPD) game can provide evidence of our findings. The strategy Tit-For-Tat (TFT) has been postulated by Axelrod and

Hamilton [AH81] to be an evolutionary stable strategy for the iterated PD game. However, in the presence of Tit-For-Two-Tats (TfTT), a strategy with which the agent does not retaliate until there have been two successive defections, the TfT strategy is not stable. The strategy that looks at the information from the last two periods can outperform the one with only one period of historical information. On the other hand, too much memory leads to more computational costs and complication. The unnecessary complication brings in noise to the decision-making process which leads to loss of efficiency.

Second, when a community consists of agents with different memory sizes, more intelligence (longer memory in our study) does not necessarily make an agent better off. The outcome depends on the type of an agent's trading partner. It is in the best interest of an agent to have the same amount of working memory as its partners. It seems that "thinking in the way your partner thinks" is the most efficient strategy. This seems counterintuitive, since usually having more information and being more intelligent puts us in a superior position. In our experiments, after we dropped the costs for both sides, the trading stage becomes a symmetric game. In this case, it seems that the best response of one party given the strategy of the other party is to play the same strategy. So, if one agent conditions its actions on the historical information of one period, its partner should do the same thing. The buyer agent in our system is intelligent enough to pick the one with the same amount of memory when it is possible and achieve an efficient outcome, and to be indifferent when the two bidder agents' memory sizes are different from that of the buyer agent.

6 Conclusion

In this paper, we presented an Exchange Game consisting of sequential and simultaneous games. Such a mixed game grants the players choices of entering a game and selecting a partner. Players face costs, a factor that can represent the risk of playing the game. This game is not proposed as a true description of the auction or exchange process in e-marketplaces, but rather as a useful approximation. We created a three-agent system to study the impact of memory size under different cost schemes. The experiments discussed here are warm-up experiments to provide initial demonstration of the importance of working memory in system design.

We found that the relationship between working memory and the efficiency of the system has an inverted U-shape, i.e., a seemingly optimal level of memory size presents itself. When we mixed agents with different memory sizes together, agents with the same amount of working memory generate the most efficient outcome. These results support the importance of working memory in agent and system design, as well as in theoretical modelling of decision-making when historical information is involved. The optimal mem-

ory size we found from our experiments coincides with the action space of the trading game. However, whether the optimal level of working memory is a function of the action space is not clear. It will be interesting to conduct further research on finding the optimal amount of working memory and the determinants of the optimum. In these experiments, agents' working memories are fixed and they cannot adjust it through learning. In the future research, it will be interesting to see what will happen if memory size is included in agents' policy spaces.

7 Acknowledgements

The work reported here was supported in part by NSF grant number SES-9709548. The author also gratefully thanks Dr. Steven O. Kimbrough and Dr. D.J. Wu for their insightful comments and discussion.

A Parameter Values

Parameter	Value
τ_1	6 (0.999992^{n1})
τ_2	6 (0.99997^{n2})
τ_3	6 (0.999^{n3})
Learning rate α	0.2
Discount factor γ	0.95
T	0.6
B	0.4
S	0.1
P	0.2
Number of iterations in each run	400,000
Number of runs in each experiment	100

References

[AH81] Robert Axelrod and W.D. Hamilton, *The evolution of cooperation*, Science **211** (1981), 1390.

[BS00] Adam M. Brandenburger and Harborne (Gus) W. Stuart, *Biform games*, Working paper No. 00-06, Harvard NOM Research Paper, http://ssrn.com/abstract=264199, November 2000.

[CB98] Caroline Claus and Craig Boutilier, *The dynamics of reinforcement learning in cooperative multiagent systems*, Proceedings of the Fifteenth National Conference on Artificial Intelligence (Menlo Park, CA), AAAI Press/MIT Press, 1998, pp. 746–752.

[Del03] Chrysanthos N. Dellarocas, *Efficiency and robustness of binary feedback mechanisms in trading environments with moral hazard*, Working Paper No. 4297-03, MIT, Sloan School, Cambridge, MA, January 2003, http://ssrn.com/abstract=393043.

[HW98] J. Hu and M. P. Wellman, *Multiagent reinforcement learning: Theoretical framework and an algorithm*, Fifteenth International Conference on Machine Learning, July 1998, pp. 242–250.

[KL97] Steven O. Kimbrough and Ronald M. Lee, *Formal aspects of electronic commerce: Research issues and challenges*, International Journal of Electronic Commerce **1** (1997), no. 4, 11–30.

[KLK04] Steven O. Kimbrough, Ming Lu, and Ann Kuo, *A note on strategic learning in policy space*, Formal Modelling in Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, 2004.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KWZ02] Steven O. Kimbrough, D. J. Wu, and Fang Zhong, *Computers play the beer game: Can artificial agents manage supply chains?*, Decision Support Systems **33** (2002), no. 3, 323–333.

[SB98] Richar S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, The MIT Press, Cambridge, MA, 1998.

[SC95] T. Sandholm and R. Crites, *Multiagent reinforcement learning in iterated prisoner's dilemma*, Biosystems **37** (1995), 147–166, Special Issue on the Prisoner's Dilemma.

[Sha89] C. Shapiro, *The theory of business strategy*, Rand Journal of Economics **20** (1989), 125–137.

[Sut91] J. Sutton, *Sunk costs and market structure: Price competition, advertising, and the evolution of concentration*, The MIT Press, Cambridge, MA, 1991.

[WD92] C.J.C.H. Watkins and P. Dayan, *Q-learning*, Machine Learning **8** (1992), 279–292.

[ZKW02] Fang Zhong, Steven O. Kimbrough, and D. J. Wu, *Cooperative agent systems: Artificial agents play the ultimatum game*, Journal of Group Decision and Negotiation **11** (November 2002), no. 6, 433–447.

Investigations of Granularity and Payoffs in 2×2 Games under Replicator Dynamics

Sofia Chajadine[1], Daniel Mack[2], and Aaron Jeffrey Slan[3]

[1] University of Pennsylvania, Philadelphia, PA USA,
chajadis@wharton.upenn.edu
[2] University of Pennsylvania, Philadelphia, PA USA,
slan@wharton.upenn.edu
[3] University of Pennsylvania, Philadelphia, PA USA,
danieljosephmack@yahoo.com

Abstract. This paper describes an investigation of several 2×2 games in iterated form. Players play the games repeatedly and are limited to mixed strategies, with particular actions chosen probabilistically. The games investigated include Prisoner's Dilemma, Chicken, and Stag Hunt in various forms. The reward structure and the granularity of the games—number of games played per generation in the replicator dynamics—are the main factors investigated, with surprising results.

1 Introduction

This paper describes results of experiments on several simple games under replicator dynamics. The experiments simulate interaction within a population of relatively basic artificial agents, each defined entirely by the strategy it is programmed to carry out. The game is played for multiple generations, after each of which the proportion of every strategy is adjusted to reflect its performance relative to other strategies. If a strategy performs very poorly, it may go extinct. Similarly, there may be cases in which a strategy performs so well that it takes over the whole population.

The games that we have chosen to experiment with are all 2×2 games. Such games involve two players, who—on each round of play—can choose between two courses of action and who receive payoffs that are dependent on both their own and their counter-player's decisions. We experimented with, and report on, the Prisoner's Dilemma game, the Stag Hunt game, and variant forms of Chicken.

While our experiments on Prisoner's Dilemma tried to replicate previous work by Nowak and Sigmund [NS92], most of our other experiments involve modifications of various game and simulation parameters. We examined the number of games played per generation, representing the level of *granularity* for the simulation, to determine whether it had a significant effect on the results. We also tested the robustness of the payoff structure by varying the

relative magnitudes of payoffs while remaining within the broad definition of the game in question.

The remainder of the paper discusses several interesting results. The emergence of Generous Tit for Tat (GTFT) strategies in stochastic Iterated Prisoner Dilemma (IPD) is highly dependent on the granularity of the simulated environment, i.e., on the number of games played per generation in the replicator dynamics. Further, we found that in all three games considered—Chicken, IPD, and Stag Hunt—well-accepted findings did not withstand radical changes to the payoff structure. Also, Flag Chicken simulations demonstrated that strong group allegiance and outsider prejudice can evolve in the presence of an arbitrary identifiable feature. This finding remains robust even with uncertainty in the identification process.

To explain the experiments as clearly as possible, we proceed as follows. First, §2 presents the games, along with remarks regarding application to problems in the real world. A detailed discussion of the simulation methodology follow in §3. We give special attention to the propagation rules used and specific variations of the games implemented. Finally, we discuss the principal findings of our experiments in §4.

2 The Games

2.1 Prisoner's Dilemma

Description. After the now-standard treatment in [LR57], two criminals have been arrested and thrown into separate cells, so they cannot communicate with each other. The candid prosecutor explains to each of them their identical situations. If they both confess to having committed a crime, they will both go to jail for three years. If they both maintain their innocence, they will only stay in jail for one year. However, if one defects and testifies against his partner, the defector will walk out free while his partner in crime gets five years of imprisonment. Each prisoner has no way of observing the other's actions before choosing his own. In matrix, or strategic, form, the game looks like this :

Table 1. Prisoner's Dilemma. Canonically: R=Reward, T=Temptation, P=Penalty, S=Sucker.

	Player A pleads innocent (Cooperate)	Player A confesses (Defect)
Player B pleads innocent (Cooperate)	Reward, Reward (3,3)	Sucker, Temptation (0,5)
Player B confesses (Defect)	Temptation, Sucker (5,0)	Penalty, Penalty (1,1)

In addition, for a Prisoner's Dilemma, it is generally required that

$$T > R > P > S \tag{1}$$

and that

$$2R > T + S \tag{2}$$

The Prisoner's Dilemma is therefore "a conflict between personal rationality and mutual risk" [Sky01]. Each player's payoff ultimately depends on what the other player decides to do. Weighing the two courses of action available to them, and being rational in the received sense, both players realize that defecting (confessing) is the best decision, regardless of what the other player chooses to do. If player A confesses, player B is better off confessing as well. If player A pleads innocent, then player B avoids jail entirely by confessing.

Iterated Prisoner's Dilemma. While defection appears to be the best strategy if the scenario occurs only a single time (and classical game theory agrees), the choice becomes more complicated if the situation is repeated many times by the same two players, as it is in the case of the iterated prisoner's dilemma (IPD). If one defects now, he has little (or at least reduced) expectation that his opponent will cooperate in the future. Therefore, the prospect of future interactions raises the possibility of cooperating in the present in order to signal a willingness to do so in the future, and thereby possibly gain that cooperation.

Robert Axelrod conducted two IPD tournaments. They involved a variety of strategies with names such as Tit for Tat, All Defect, SuperRetaliate, or Random. Successful strategies were found to have a set of common characteristics: they are nice (meaning they never defect on the first move), send clear signals as to the behavior their opponent can expect, reciprocate both cooperation and defection, and are not envious (do not try to do better than the other strategies but seek to maximize their payoffs instead) [Axe84, pages 109-123].

Tit for Tat (TFT), a simple strategy that reciprocates the opponents last move, was found to be the strongest strategy. However, TFT does not work as well in noisy environments where a competitor's move may be misinterpreted, principally because TFT will not be generous enough and will tend to get into costly spirals of defection.[1]

[1] [NS92,NS93]

2.2 Chicken

Description. The game of chicken was brought to popular attention in the James Dean movie *Rebel without a Cause.* The game is simple: two players get in their cars and start speeding towards a cliff. The point of the game is to show bravado by being the last driver to stop before self-destruction.

Consider the payoffs. Suppose A swerves first while B stays steadfast. B gets a higher payoff (W for winner) than A (C for Chicken). If they both swerve at the same time, they get an equal payoff (E for equal), just as when they both do not swerve (D for demolition or death). We thus arrive at the following payoff distribution, represented in matrix format, as in Table 2.

Table 2. Chicken. Canonically: W=Winner, E=Equal, C=Chicken, D=Demolition

	Player A swerves	Player A is steadfast
Player B swerves	Equal, Equal (1,1)	Chicken, Winner (0,5)
Player B is steadfast	Winner, Chicken (5,0)	Demolition (-5,-5)

It is generally required that

$$W > E > C > D \tag{3}$$

and

$$2E < W + C \tag{4}$$

so it pays to take turns swerving.

Applications. This game is of special interest to the international relations, international political economy and business communities. For example, consider the situation in which two countries are negotiating a trade treaty. The country that stays steadfast in its position and succeeds in the other making concessions will receive the higher payoff. If they compromise the payoff to each country will be situated somewhere in between the two afore-mentioned payoffs. If, however, they do not manage to sign a treaty and get locked up in a trade war, the effects on both countries would be detrimental. The essence of this game is that, while you are best off if your cowardly opponent swerves first, there is an overwhelming incentive to swerve before the cliff, as riding off the cliff is the worst of all possible outcomes for both players. In this sense, Chicken is fundamentally different than a one round PD, in which defection is clearly optimal.

2.3 Stag Hunt

Our third game is known as Stag Hunt, and was originally described in Rousseau's *Discourse on Inequality* as follows (cited in [GMS98]):

> If it was a matter of hunting a deer, everyone well realized that he must remain faithful to his post; but if a hare happened to pass within reach of one of them, we cannot doubt that he would have gone off in pursuit of it without scruple...

In other words: The hunters have the choice of hunting either stag or hare. One hunter cannot hunt stag alone successfully. A hare is less valuable than a stag. A hunter's probability of getting a hare is independent from the choices of the other hunters. The more hunters there are, the higher the probability of succeeding in capturing or killing a stag.

The Stag Hunt game comes in various forms. The one we investigate may be attributed to Hume and his story of the rowers:

> Two men who pull at the oars of a boat, do it by an agreement or convention, tho' they have never given promises to each other..." [Hum78].

Here the men could choose to row or not to row. Only when the two men row can they get to their destination. If one man decides to row and the other does not row, the boat does not move forward. The worst outcome for oneself is to row alone, because one is wasting energy while moving in circles. Following this line of thought, it is a middle outcome when one chooses not to row, regardless of whether the other man rows or not.

We can see that in Stag Hunt, unlike the Prisoner's Dilemma, there is no dominant strategy. Each hunter's optimal strategy depends on what he believes the other hunters will decide, and both stag hunting and hare hunting are equilibria.[2]

In matrix form, the stag hunt game can be represented as follows:

As is customary, we require that

$$R > T > P > S \tag{5}$$

The player finds himself split between considerations of mutual benefit and considerations of personal risk. The problem of trust makes the player think that the other players will defect, often pushing the result to the hare hunting equilibrium.

[2] Note that an Iterated Prisoner's Dilemma game is similar in some ways to a two-person Iterated Stag Hunt. Because the shadow of the future makes the players' moves in the present round matter for the next, cooperation may be more desirable yet riskier than defection.

Table 3. Stag Hunt. Canonically: R=Reward, T=Temptation, P=Penalty, S=Sucker

	Player A hunts stag	Player A hunts hare
Player B hunts stag	Reward,Reward (4,4)	Sucker,Penalty (0,3)
Player B hunts hare	Penalty,Sucker (3,0)	Penalty,Penalty (2,2)

3 Methodology

In our replicator dynamics simulations, pairs of strategies were repeatedly drawn at random from a population. (Initially, all strategies were drawn with equal probabilities and with replacement.) The strategies played a *round* of games (set by the parameter `GamesPerRound`) against each other and the resulting scores were recorded. This—random drawing of two strategies and subsequent play of a round of games—was repeated a certain number of times, set by the parameter `RoundsPerGeneration.` Play of `RoundsPerGeneration` rounds constituted one generation of play. After one generation's worth of play, the likelihood of each strategy being drawn, which represents the strategy's prominence in the population, was adjusted to reflect the strategy's performance in the current generation. The process of competition and adjustment is repeated for a number of generations, represented by the parameter `NumberOfGenerations`.

3.1 Number of Generations

The number of generations used for the simulations ranged from 2,000 to 12,000. The choice of number was made with the intent that long-term trends be evident and observable. Often this meant choosing enough generations to allow an apparent equilibrium mix of strategies to develop. Simulations in which no stable pattern of evolution emerged were rerun with a greater number of generations.

3.2 Rounds Per Generation

The number of rounds per generation was varied experimentally to examine the impact of granularity upon outcome. The results are discussed below.

3.3 Games Per Round

The expected value of `GamesPerRound` was fixed in our experiments. It was normally distributed with mean 80 and standard deviation 20 (with negative values discarded). The game length was thus not available to the strategies.

3.4 Propagation Rule

The formula for adjusting the proportion of a strategy x from generation n to generation $n+1$ reflects both x's current proportion and its performance in generation n:

$$w_{x,n+1} = w_{x,n} \times \frac{\overline{P}_x}{\overline{P}_{all\ \ strategies}} \tag{6}$$

where w represents the weight (or probability of selection) and $\overline{P}$ represents the average per round payoff of all games played.[3]

3.5 Iterated Prisoner's Dilemma

Stochastic Prisoner's Dilemma, Finite Length. Single frame reactive strategies were represented in the form (i, c, d), where i represents the probability of initial cooperation, c the probability of cooperating after observing a counter-player cooperation, and d the probability of cooperating after observing a counter-player defection. Deterministic strategies were outlawed, as a small level of uncertainty (1%) was assumed to be intrinsic to the interactions. Pairs of strategies competed in games of unknown, finite length.

The initial population of strategies consisted of two classes of strategies: those who cooperated in the first round with probability of 0.99, and those who defected with the same probability. Within each class, 121 (c, d) pairs where uniformly distributed on the unit square. All strategies began with equal weights of occurrence. Payoffs for the game were varied as an experimental parameter.

Stochastic Prisoner's Dilemma, Infinite Length. In the special case of games of infinite duration, the initial move has no effect on the expected outcome when no strategy acts deterministically. Therefore, one-frame reactive strategies of the form (c, d) where considered, where c and d have the same meanings as above (cooperate and defect). To approximate an iterated game of infinite duration, expected payoffs, rather than observed payoffs were used. The expected payoff of a strategy $s_1 = (c_1, d_1)$ when playing against strategy $s_2 = (c_2, d_2)$ are as follows :

$$V(s_1|s_2) = 1 + 4 \cdot t' - t - t \cdot t' \tag{7}$$

where $V(s_1|s_2)$ is the value of playing strategy s_1 against strategy s_2, and

$$t = \frac{\{d_1 + (c_1 - d_1) \cdot d_2\}}{\{1 - (c_1 - d_1) \cdot (c_2 - d_2)\}} \tag{8}$$

[3] We note that the formula may be modified to accommodate discounting, but we chose not to explore this.

and

$$t' = \frac{\{d_2 + (c_2 - d_2) \cdot d_1\}}{\{1 - (c_2 - d_2) \cdot (c_1 - d_1)\}} \tag{9}$$

(See [NS92] and [GMS98, page 169].)

The initial population used for simulations was 121 (c, d) pairs uniformly disturbed on the unit square (but not at 0 or 1). Again, deterministic strategies were outlawed and equal weights were initially assigned.

3.6 Chicken

Chicken with a Temporal Element. Stochastic Temporal Chicken is a variant of chicken where players have three opportunities to swerve. If they see their opponent swerve first, they automatically are accorded victory. Strategies are of the form (P[swerve at first opportunity], P[swerve at second opportunity], P[swerve at final opportunity]). (Note: P[X] = the probability of X occurring.) These probabilities are bounded to the range [.1, .9] to reflect the uncertainty inherent to the game. Even with a high probability of swerving, an agent may still crash. Strategies play a single round in each game. The exact payoffs they received were varied as an experimental parameter.

Flag Chicken. In Stochastic Flag Chicken, players first raise a flag to identify themselves as members of either the green or the red group. They then play a one-game round of traditional chicken, with payoffs: W=6, E=4, C=3, and D=0. Strategies are of the form (P[raising green flag], P[swerve after seeing a green flag], P[swerve after seeing a red flag]). (Note: P[raise red flag] = 1-P[raise green flag].) In this way, strategies are allowed to discriminate based upon perceived group affiliation. As misinterpretation of the flag and imperfect driving are assumed to be inherent; all probabilities are bound to the range [.1, .9].

3.7 Stag Hunt

In Stag Hunt, strategies are of the form (P[hunting stag], P[hunting stag | other player hunted stag in previous round], and P[hunting stag | other player hunted hare in the previous round].) Payoffs were varied as an experimental parameter. The base payoffs we used are Reward = 4, Temptation = 3, Punishment = 2 and Sucker = 0.

4 Discussion of Findings

4.1 Granularity Thwarts Generosity in Stochastic IPD

One of our main goals was to test the validity of the modeling system by reproducing published results for stochastic IPD. In "Tit for Tat in Heterogeneous Populations" [NS92] Nowak and Sigmund report that, using stochastic

strategies identical to ours and the regime described in §3.5, generous retaliatory strategies eventually dominate TFT and control the population. These GTFT strategies—$(0.99, 0.3)$ is the eventual winner—are driven even closer to extinction than TFT by the early onslaught of defectors. However, they recover in later generations and are able to dominate in a noisy world of cooperators.

Our initial results were to the contrary. As with Nowak and Sigmund, defector strategies went on an early rampage, capturing nearly half of the population. TFT, bruised but not beaten, fights back and recovers robustly. However, TFT was not merely the pivot Nowak and Sigmund describe. TFT rules the land, controlling 100% of the population. AllD (Always Defect) rises prominently in the early rounds, conquering half the population (the other half being held mostly by fellow defectors). Around the 250^{th} generation, naïve cooperators die out and TFT makes its move. Over the course of 60 generations, it goes from near extinction to total domination. See Figure 1.

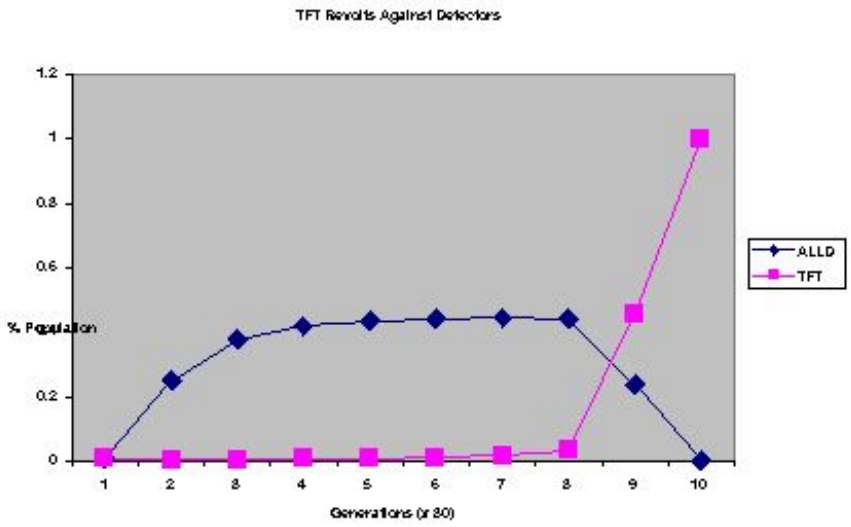


Fig. 1. TFT Revolts against Defectors

The reason more generous strategies do not come to prominence is that they go extinct during the era of the defectors. Unlike Nowak and Sigmund's model, where average fitness for the generation is calculated as an expectation, strategies in our model are actually drawn at random and paired against each other. Consequently, extinction can and does occur. If a strategy is not drawn at all in a particular generation, it simply dies out. So it seems the GTFT's success hinges upon the granularity of the model.

To test this hypothesis, simulations were run with less coarse models. At a granularity level of 300,000 rounds per generation, a GTFT strategy of $(0.99, 0.1)$ finally survived the defector era and triumphed over TFT. In doing so, it lingered near extinction for hundreds of generations. At one point, the likelihood of drawing GTFT was roughly 2 in 100,000. In the simulation with 300,000 games per round, TFT first wages war against defectors. As the population becomes filled with nice strategies, GTFT rises from obscurity to conquer its less generous relative.

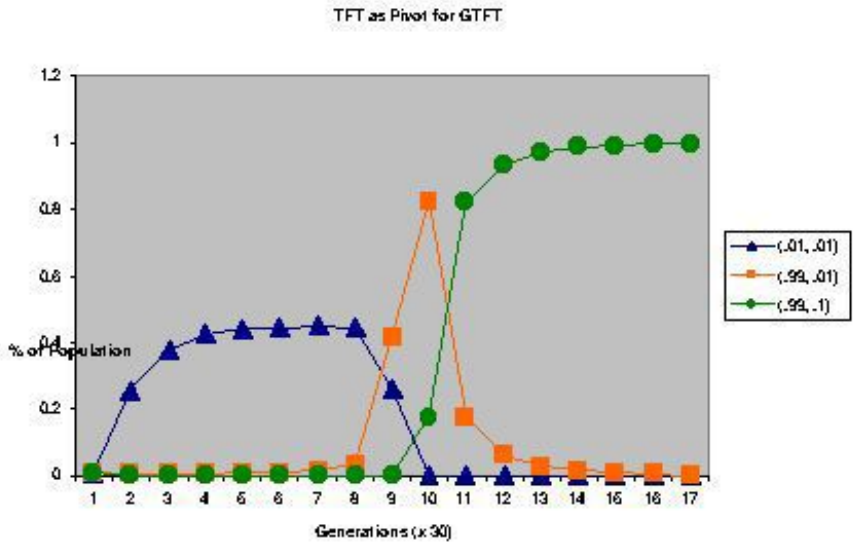


Fig. 2. TFT as Pivot for GTFT

The degree to which granularity impedes the emergence of generosity is not only a problem for modeling, but also a potentially strong barrier for real world generosity. We note that Skyrms also comments that less granular populations are more likely to fall into polymorphic traps [Sky96]. Our results predict that among coarse populations, say the nation states of the world, generosity is a recipe for extinction. Generous nations may be so vulnerable to attack that they are unable to sustain themselves and spread their ideology of generosity. On the other hand, if one looks at cooperation at the level of single genes,[4] where the population is more properly modeled by a continuous idealization, the prospects for generosity seem much stronger.

[4] Dawkins [Daw89] looks at competition at the gene level, concluding that selfishness increases fitness. However, the whole human organism can thought of as a cooperation among thousands of individual genes.

4.2 Temporal Chicken Sensitive to Risk Preference

Temporal Chicken was an attempt to see if granting agents timing options would make a difference in the kinds of strategies that would thrive in an ecological simulation (i.e., under the replicator dynamics). To get a better understanding of the influence timing can have we conducted simulation with three different payoff structures. While there were naturally similarities among all three simulation types, it is striking how changing the payoffs significantly changed the relevance of the timing decision. In the following we will discuss the results, starting with the initial payoff structure of W=6, E=4, C=3, D=0, then proceeding with the one that makes demolition much worse to live with and finally one where the difference between winning, losing or being part of a draw become insignificant in view of the huge penalty involved in demolition.

Normal Payoffs and the Siege of the Last Minute Strategies. We define normal payoffs as W=6, E=4, C=3, D=0, values commonly appearing in the literature. The striking outcome of the simulation here is that only strategies that have the lowest possible probability of swerving in the first two time periods survive. In fact all other strategies died off before 850 rounds were played. Furthermore, there are no clear winners or any other patterns that are distinguishable among the surviving strategies. This outcome can be interpreted as meaning that if there is a somewhat equitable difference among the different payoffs, it pays to wait until the last minute before deciding to swerve. Using this strategy, one will win against all strategies that have a tendency to swerve in the first two time periods. These gains seem to allow for last minute strategies to gradually take over more and more of the population. Thus, they are leaving all other strategies with an increasing probability of incurring a loss and giving up the winning payoff. This seems to outweigh the increased probability of the last minute strategies meeting themselves and once in a while incurring complete demolition.

Costly Demolition and the Reign of the Generous Last Minute Strategies. When we changed the payoffs to W=1000, E=900, C=850, D=0 we were expecting this to help non-last minute strategies to survive. Although they did not make it in the end, they now survive past round 2250. However, the main difference that the change in payoffs produced was that only three last minute strategies survived and that there was a clearly dominating strategy.

Specifically, the most generous last minute strategies survived, namely, the ones that have a high probability of swerving, albeit only in the final time period. In fact, the last minute strategies died out in order of their generosity, with the first five[5] dying out while there were still non-last minute strategies

[5] (0.1,0.1,0.1), (0.1,0.1,0.2), (0.1,0.1,0.3), (0.1,0.1,0.4), (0.1,0.1,0.5)

in existence and (.1,.1,.6) surviving until after the 3900th round. Further, the clear predominance of the most generous last minute strategy shows how the payoff distribution made it very important not to be suckered out in the first two rounds but still have a very low probability of actually facing the cost of demolition; two characteristics that (.1,.1,.9) combines best out of all the strategies.

Costly Demolition, Minimal Winner's Advantage and the Survival of Many. Changing the payoffs to W=1000, E=999, C=998, D=0 allowed 136 different strategies to survive until the end of the simulation. This statement of course should be qualified by the fact that the 10 best strategies made up almost 85% of the population by the end, the first 25 made up 95% and that the least successful 71 strategies together only made up 1% of the entire population.

The most striking change that this variation in payoffs brought about, is that the dominance of the last minute strategies was clearly broken. Not one of them was in the top ten. In turn the top ten strategies were entirely made of agents that had probabilities of between 0.8 and 0.9 of swerving in the two last time periods and 0.1 and 0.4 for the first time period. Here it is especially interesting to look more closely at the probabilities of the three most successful strategies: (.1,.9,.9), (.2,.9,.9), (.1,.8,.9)

In summary, it seems pivotal to avoid the punishing demolition payoff; the reward of bragging rights that comes with winning has become insignificant in comparison to the perils of demolition. Therefore, successful strategies are only willing to risk the first round for a chance to win against ones opponent. After that, they must minimize their chances of never swerving to make sure that demolition is avoided. Again, successful strategies have been shown to be not only the product of the general game, but also of the particular payoffs applied.

4.3 Flag Chicken

In daily interactions, one is likely to make countless associations, ranging from race and ethnicity to tone of voice or table manners. Attached to these associations are heuristics that guide both one's actions and expectations towards others. In the traditional Chicken game, players have no way of associating themselves with a certain group or determining if the other player is part of a group. The Flag Chicken experiment allows the players to identify themselves to the other player as red or green.

By allowing for the possibility of discrimination, we hypothesized that discriminable, easily identifiable agents would become more prevalent in the population. We were not sure what would happen to unidentifiable agents, those who have almost the same chance of signaling red or green.

As an illustration, examine one typical run of the simulation. Strategies that raised the green flag with 90% certainty (the most allowed) took over the

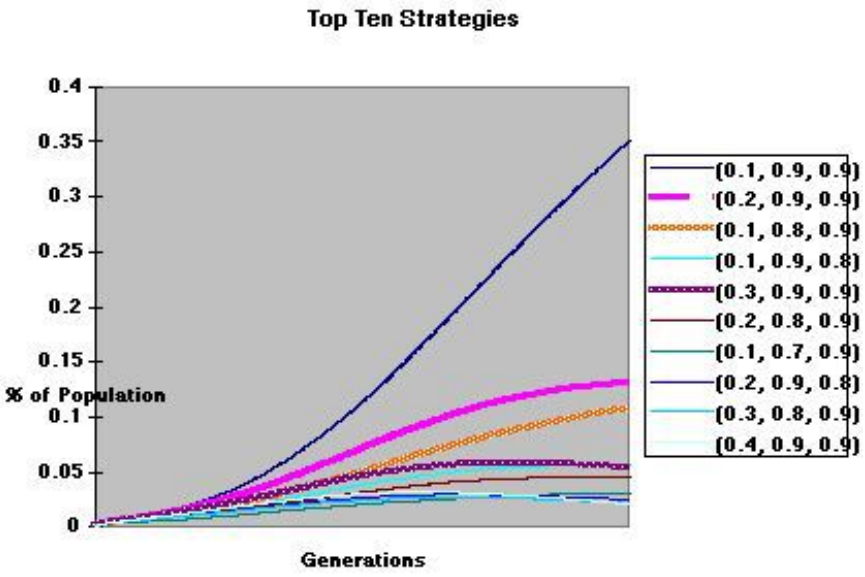


Fig. 3. Top Ten Strategies. The evolution of the ten strategies that are most successful at the end of the simulation. With the extreme payoff structure implemented here, it is interesting to see how avoiding demolition has become very important and allows for a different set of strategies to thrive.

population, representing 100% of the population by the end of the simulation. The green strategies that did best were those that courteously swerved when facing another green player, but rudely refused to swerve when facing a red player. The top performer was strategy (.90, .70, .20), which ended the simulation at 14.62% of the population. Other successful strategies included (.90, .02, .70) with 6.43%, (.90, .90, .20) with 7.99%, (.90, .79, .10) with 5.83%, (.90, .90, .10) with 4.84%, and (.90, .79, .50) with 3.49%. Not swerving when facing another green seems to pay off when done moderately, as can be observed from our data. This can be explained by the competing temptation to battle strategies within ones own group.

The indecisive strategies, which have mid-range probabilities of signaling signal red or green are wiped out quickly, usually before the tenth generation. The red strategies also disappeared very quickly, without having been able to team up and reap the benefits of in-group cooperation. What seems to be happening is that players from the winning, strong identity group (here the greens) are in fact able to benefit more from cooperation with their peers in early rounds by pure luck. This initial advantage allows that group to increase more quickly and to even further perpetuate their strength. The other players (here the red), quickly decrease in the population, and therefore find fewer

opportunities to cooperate with other reds. They are quickly eliminated and only one group prevails (here the green). The fact that the green survived in this instance was mere chance, and it was equally likely that red could take over.

The result—that both indecisive strategies and strategies that align to the wrong color die out fairly quickly, thus leaving a homogeneous group of same-signaling strategies to make up the population—also holds under changing granularity assumptions.[6] However, differences do arise in terms of the number of strategies that survive and in the distribution of the percentage of the population made up by each of the strategies. In our run with 15,000 games per generation 49 Strategies survived until the cut off; with 30,000 games per generation 56 survived; and with 150,000 and 300,000 all 81 clearest signaling strategies survived. Then, if we look at the standard deviations of the percentage of the population made up by each of the strategies we see that the 15k run resulted in an ending standard deviation of 0.007226, the 30k run, 0.007055, the 150k run, 0.004852 and the 300k run, 0.004517. These differences could be further magnified by not including the non-decisive strategies and the wrong alliance ones into the calculation of the standard deviation. Thus, granularity results in more strategies surviving and a reduction in the difference in success among these surviving strategies.

It is natural to speculate on the parallels with discrimination by a group against another in real life. Our experiment demonstrates that the ability to identify members of a group and the willingness to treat others differently (not necessarily better or worse) are sufficient conditions to cause the extermination of whole groups. Those who rise to the top are the ones whose behavior towards foreign groups is particularly hostile, and whose behavior towards their own group particularly benign. Always doomed are the people who claim little allegiance to a particular group, and are not easily identifiable.[7]

For these reasons, we speculate that it would be very hard for peaceful coexistence to arise between two populations that discriminate against each other when members of one group regularly interact with members of the other. This may explain all sorts of spatial segregation: segregation by ethnic background in neighborhoods, by gender in the workplace or even in social interactions.

4.4 More on Changing the Payoffs: IPD and Flag Chicken

In order to investigate the effect of changing the Prisoner's Dilemma payoffs, we constructed a simulation that, at the onset, was very unfriendly toward cooperation. Round lengths were short, averaging 50 games, and with only

[6] Wrong with SC (Flag or Signal Chicken) means not belonging to the arbitrarily chosen strong identity group: either red or green.

[7] These are the strategies that signal red or green roughly half the time.

15,000 rounds played per generation, extinction was common. The simulation was first run with the standard PD payoffs of T=5, R=3, P=1, and S=0. Defectors killed off all strategies, including TFT, in less than 300 generations. The experiment was then repeated with of T=1000, R=999, D=10, and S=0.[8] With these new payoffs, not even TFT was forgiving enough to survive. The population was dominated by strategies similar to AllC (cooperation everywhere).

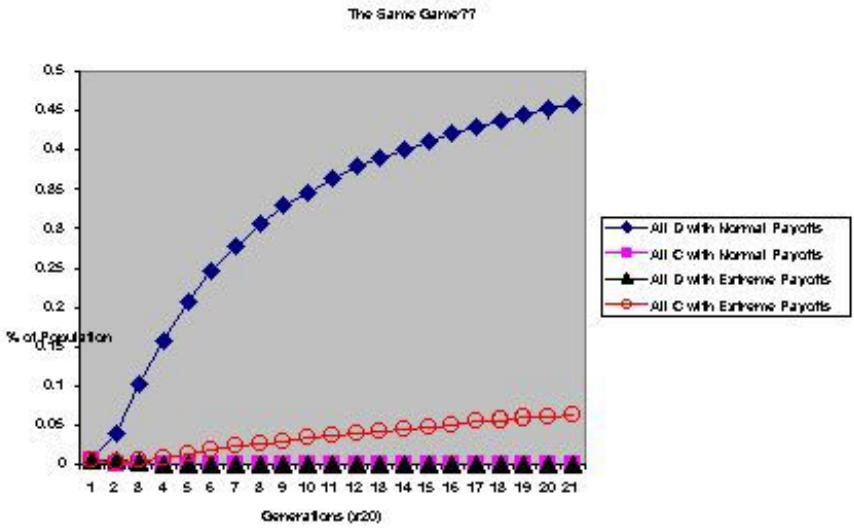


Fig. 4. PD under Two Different Payoff Regimes. This graph shows a comparison of AllD with AllC over the first 200 generations in the two different payoff scenarios. With the normal payoff scheme, AllD dominates early while AllC is rapidly driven to extinction. When the payoffs are changed to the extreme scenario, it is AllD that is wiped out on the onset, while AllC ascends slowly over time.

Although both scenarios are true PDs, the results they produce are so drastically different that it seems misleading to refer to them as the same game. This experiment demonstrates that the well-established results of Axelrod and others do not apply to all games that fit the strict definition of a PD. A more precise understanding of the sensitivity of these results to the payoffs used remains to be explored, but our small experiment shows that broad generalizations about the PD should be made with care.

The hypothesis that not only relative order, but also relative magnitude of payoffs can shape the fundamental nature of these simple, 2×2 games is

[8] Note that T>R>P>S and R>(T+S)/2, fulfilling the traditional requirements of a PD game.

confirmed by similar experiments involving the Flag Chicken game. As the previously mentioned experiments have shown, Flag Chicken simulations typically evolve to a population in which all strategies tend to raise the same color flag, and punish any strategy that defiantly raises a different one. However, in the extreme payoff example, where crashing is catastrophic compared to the slight humiliation of being a swerver,[9] discriminatory strategies are done in. The strategies that survive are those that cautiously swerve, regardless of the flag they show or see. Once again, fundamental emergent properties of the game vary under different payoffs.

These experiments demonstrate what on second thought seems obvious: that hopes for cooperation lean heavily upon the costs of chaos. If actors are in an environment where the benefit from successfully cheating counterplayers is miniscule in comparison to the costs of widespread defection, wise actors will be inclined to cooperate.

4.5 Stag Hunt

Our first Stag Hunt simulation, Normal Stag Hunt, features the base payoffs described in Table 3. In addition to Normal Stag Hunt, we ran three Stag Hunt simulations with varying payoff structures. In Extreme Stag Hunt, we set Reward = 700, Temptation = 500, Punishment = 200 and Sucker = 0. This increased the gap between the Reward and Temptation payoffs on one part and the Punishment and Sucker payoffs on another. The third simulation, Very Extreme Stag Hunt (Reward = 1000, Temptation = 200, Punishment = 50 and Sucker = 0) further accentuated this gap, and increased the Reward-Temptation gap. Finally, in Clustered Stag Hunt, we clustered the three highest payoffs, with Reward = 1001, Temptation =1000, Punishment = 999 and gave the sucker a payoff of 0.

The results of the simulations are quite compelling. In all test runs we did, the trends were clearly recognizable. In the first three simulations, the most successful strategy was (.99, .99, .99), which basically hunts stag all the time no matter what the other player is doing. The only times that this strategy hunts hare are due to the non-deterministic nature we implemented in all strategies. However, depending on the relative magnitude of the payoffs, complete eradication of all other strategies can take more or less time. The more extreme the payoffs, the faster the strategy's takeover of the population. The next most successful players are those that only reduce their probability of hunting stag when their counterpart hunts hare.

It is nonetheless interesting to note that in all three simulations, those strategies that do not give the benefit of the doubt and hunt hare in the first round did significantly worse as a group but within themselves had the same relative survival rate as the parallel strategies that hunt stag in the first round. The most successful of these strategies is (.1, .99, .99), i.e., the one that

[9] To be specific, payoffs of W=1000, E=900, C=850, D=0.

always hunts stag except for the first round. The results of the first three Stag Hunt simulations, very much like the Prisoner's Dilemma simulation results, speak in favor of the more generous strategies, and confirm the advantages of cooperation in a noisy stochastic environment.

However, clustering the three highest payoffs can totally discourage cooperation. Under this scenario, all the strategies that are forgiving (those that sometimes hunt stag after the opponent hunted hare) die out very quickly. Then there are two non-forgiving categories: the initial hare hunters, and the initial stag hunters. Within these two categories, the success of the non-forgiving strategies is correlated to the middle percentage. The higher the probability of hunting hare after the opponent hunted stag, the better off the strategy. Accordingly, the most successful strategy is (.01, .01, .01).

An interesting fact that we observed is that the (.99, .01, .01) strategy is considerably less successful and even dies out before the end of the simulation, although the only difference between it and the most prevalent strategy is its first move choice.

Similar to the results we had for IPD, Temporal and Flag Chicken, we can conclude that relative magnitude of payoffs changes the Stag Hunt characteristics dramatically. Here, we went from a game where generosity and cooperation performed best all the way to a game where defectors dominated the population of agents.

5 Acknowledgements

The work reported here was supported in part by NSF grant number SES-9709548.

A Summary of Simulations

See Table 4. In all simulations, strategies were selected from the unit cube (or square) with uniform random deviates (excluding 0 and 1).

References

[Axe84] Robert Axelrod, *The evolution of cooperation*, Basic Books, Inc., New York, NY, 1984.

[Daw89] Richard Dawkins, *The selfish gene*, Oxford University Press, New York, NY, 1989.

[GMS98] Patrick Grim, Gary Mar, and Paul St. Denis, *The philosophical computer: Exploratory essays in philosophical computer modeling*, The MIT Press, Cambridge, MA, 1998.

[Hum78] David Hume, *A treatise of human nature*, Oxford: Clarendon, Oxford, UK, [1739] 1978, Selby-Bigge, ed.

Table 4. Summary of Simulations

Experimental Parameter	Game	Game Format	No. of Gen's	Games per Gen.	Payoffs [CC, CD, DC, DD]	No. of Strategies
Granularity	IPD	Expt. Payoffs	12000	300000	[(3,3), (0,5), (5,0), (1,1)]	121
Granularity	IPD	Expt. Payoffs	12000	150000	[(3,3), (0,5), (5,0), (1,1)]	121
Granularity	IPD	Expt. Payoffs	12000	50000	[(3,3), (0,5), (5,0), (1,1)]	121
Granularity	IPD	Expt. Payoffs	12000	30000	[(3,3), (0,5), (5,0), (1,1)]	121
Granularity	IPD	Expt. Payoffs	12000	15000	[(3,3), (0,5), (5,0), (1,1)]	121
Granularity	FlagChicken	Real Payoffs	3000	300000	[(3,3), (2,4), (4,2), (1,1)]	729
Granularity	FlagChicken	Real Payoffs	3000	150000	[(3,3), (2,4), (4,2), (1,1)]	729
Granularity	FlagChicken	Real Payoffs	3000	50000	[(3,3), (2,4), (4,2), (1,1)]	729
Granularity	FlagChicken	Real Payoffs	3000	30000	[(3,3), (2,4), (4,2), (1,1)]	729
Granularity	FlagChicken	Real Payoffs	3000	15000	[(3,3), (2,4), (4,2), (1,1)]	729
Granularity	Stag Hunt	Real Payoffs	10000	200000	[(4,4), (0,3), (3,0), (2,2)]	200
Granularity	Stag Hunt	Real Payoffs	10000	20000	[(4,4), (0,3), (3,0), (2,2)]	200
Granularity	Stag Hunt	Real Payoffs	10000	2000	[(4,4), (0,3), (3,0), (2,2)]	200
Payoffs	Stag Hunt	Real Payoffs	2000	15000	[(4,4), (0,3), (3,0), (2,2)]	200
Payoffs	Stag Hunt	Real Payoffs	2000	15000	[(1000,1000), (0,200), (200,0), (50,50)]	200
Payoffs	Stag Hunt	Real Payoffs	2000	15000	[(700,700), (0,500), (500,0), (200,200)]	200
Payoffs	FlagChicken	Real Payoffs	5000	20000	[(4,4), (3,6), (6,3), (0,0)]	729
Payoffs	FlagChicken	Real Payoffs	5000	20000	[(600,600), (200,1000), (600,1000), (0,0)]	729
Payoffs	FlagChicken	Real Payoffs	5000	20000	[(900,900), (850,1000), (1000,850), (0,0)]	729
Payoffs	FlagChicken	Real Payoffs	3000	50000	[(4,4), (3,6), (6,3), (0,0)]	729
Payoffs	FlagChicken	Real Payoffs	3000	50000	[(700,700), (500,1000), (500,1000), (0,0)]	729
Payoffs	FlagChicken	Real Payoffs	3000	50000	[(999,999), (997,1000), (997,1000), (0,0)]	121
Payoffs	IPD	Real Payoffs	2000	20000	[(3,3), (0,5), (5,0), (1,1)]	121
Payoffs	IPD	Real Payoffs	2000	20000	[(999,999), (0,1000), (1000,0), (10,10)]	121

[LR57] R. Duncan Luce and Howard Raiffa, *Games and decisions*, John Wiley, New York, NY, 1957, Reprinted by Dover Books, 1989.

[NS92] M. Nowak and K. Sigmund, *Tit-for-tat in heterogeneous populations*, Nature **355** (1992), 250–2.

[NS93] ———, *A strategy of win-stay lose-shift that outperforms tit-for-tat in the prisoner's dilemma game*, Nature **364** (1993), 56–8.

[Sky96] Brian Skyrms, *Evolution of the social contract*, Cambridge University Press, Cambridge, UK, 1996.

[Sky01] ———, *The stag hunt*, World Wide Web, 2001, Proceedings and Addresses of the American Philosophical Association. http://www.lps.uci.edu/home/fac-staff/faculty/skyrms/StagHunt.pdf. Accessed September 2004.

Part V

References and Index

References

[Aal92] W.M.P. van der Aalst, *Timed coloured Petri nets and their application to logistics*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1992.

[Aal97] ———, *Three good reasons for using a Petri-net-based workflow management systems*, Information and Process Integration in Enterprises: Rethinking Documents (Toshiro Wakayama, Srikanth Kannapan, Chan Meng Khoong, Shamkant Navathe, and JoAnne Yates, eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1 December 1997, pp. 179–201.

[AB00] Alan S. Abrahams and Jean M. Bacon, *Event-centric business rules in e-commerce applications*, Workshop on Best Practices in Business Rule Design and Implementation at the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (Minneapolis, MN), OOPSLA, October 2000.

[AB01a] ———, *Event-centric policy specification for e-commerce applications*, Workshop on Policies for Distributed Systems and Networks (Bristol, UK), January 2001.

[AB01b] ———, *Occurrence-centric policy specification for e-commerce applications*, Workshop on Formal Modelling for Electronic Commerce (Norway, Oslo), June 2001.

[AB01c] ———, *Representing and enforcing e-commerce contracts using occurrences*, Proceedings of the 4th International Conference on Electronic Commerce Research, Edwin L. Cox School of Business, Southern Methodist University, Dallas, Texas, USA, November 2001.

[AB01d] ———, *Representing and enforcing electronic commerce contracts over a wide range of platforms using occurrence stores*, Fourth CaberNet Plenary Workshop (Pisa, Italy), October 2001.

[AB02a] ———, *The life and times of identified, situated, and conflicting norms*, Sixth International Workshop on Deontic Logic in Computer Science (DEON'02), Imperial College, London, UK, May 2002.

[AB02b] ———, *A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts*, Group Decision and Negotiation **11** (2002), no. 6, 487–524.

[AB02c] ———, *A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts*, Group Decision and Negotiations Journal **11** (2002), no. 6, 1–38.

[Abr02] Alan S. Abrahams, *Developing and executing electronic commerce applications with occurrences*, Ph.D. thesis, University of Cambridge Computer Laboratory, 2002.

[AEB02a] Alan S. Abrahams, David M. Eyers, and Jean M. Bacon, *An asynchronous rule-based approach for business process automation using obligations*, Proceedings of the Third ACM SIGPLAN Workshop on Rule-Based Programming (RULE'02) (Pittsburgh, USA), October 2002.

[AEB02b] ———, *A coverage-determination mechanism for checking business contracts against organizational policies*, Proceedings of the Third VLDB Workshop on Technologies for E-Services (TES'02), 2002.

[AEB02c] ———, *Mechanical consistency analysis for business contracts and policies*, Proceedings of the Fifth International Conference on Electronic Commerce Research (Montreal, Canada), October 2002.

[AEB04] ———, *Practical contract storage, checking, and enforcement for business process automation*, Formal Modelling for Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, Berlin, Germany, 2004.

[AF03] David E. Avison and Guy Fitzgerald, *Where now for development methodologies?*, Communications of the ACM **46** (2003), no. 1, 78–82.

[AG03] R. Anthony and V. Govindarjan, *Management control systems*, 11th ed., McGraw-Hill/Irwin, New York, NY, 2003.

[AGMW97] B. Adelberg, H. Garcia-Molina, and J. Widom, *The STRIP rule system for efficiently maintaining derived data*, Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997, pp. 147–158.

[AH81] Robert Axelrod and W.D. Hamilton, *The evolution of cooperation*, Science **211** (1981), 1390.

[AH94] K. A. Andersen and J. N. Hooker, *Bayesian logic*, Decision Support Systems **11** (1994), no. 2, 191–210.

[AH02] W.M.P. van der Aalst and K. Hee, *Workflow management: Models, methods and systems*, MIT Press, Cambridge, MA, 2002.

[AHL+97] W. B. Arthur, J. H. Holland, Blake LeBaron, R. G. Palmer, and P. Tayler, *Asset pricing under endogenous expectations in an artificial stock market*, The Economy as an Evolving Complex System II (Menlo Park, CA) (W. B. Arthur, D. Lane, and S. N. Durlauf, eds.), Adisson-Wesley, 1997.

[AK96] R. Ayres and P. J. H. King, *Querying graph databases using a functional language extended with second order facilities*, Advances in Databases, Proceedings of the 14th British National Conference on Databases, (BNCOD 14) (Edinburgh, UK), Springer, July 1996, pp. 189–203.

[AK02] Alan S. Abrahams and Steven O. Kimbrough, *Treating disjunctive obligation and conjunctive action in event semantics with disquotation*, Wharton Business School Working Paper Series (2002).

[Alb88] W. Albers, *Aspirations and aspiration adjustment in location games*, 1988.

[All82] L.E. Allen, *Towards a normalized language to clarify the structure of legal discourse*, Deontic Logic, Computational Linguistics and Legal Information Systems (A. A. Martino, ed.), vol. II, North-Holland Publishing Company, 1982, pp. 349–407.

[All95] James Allen, *Natural language understanding*, second ed., The Benjamin/Cummings Publishing Company, Redwood City, California, 1995.

[AM64] R J. Aumann and M Maschler, *The bargaining set for cooperative games*, Advances in Game Theory (M Dresher, L S. Shapley, and A W. Tucker, eds.), Princeton University Press, 1964.

[And58] A.R. Anderson, *A reduction of deontic logic to alethic modal logic*, Mind **67** (1958), 100–103.

[And62] ———, *Logic, norms, and roles*, Ratio **4** (1962), no. 36, 36–49.

[Aus62] John L. Austin, *How to do things with words*, Oxford at the Clarendon Press, Oxford, England, 1962.

[AV93] B. Allaz and J.-L Vila, *Cournot competition, forward markets and efficiency*, Journal of Economic Theory **59** (1993), 1–16.

[Axe84] Robert Axelrod, *The evolution of cooperation*, Basic Books, Inc., New York, NY, 1984.

[AY96] Nabil R. Adam and Yelena Yesha (eds.), *Electronic commerce: Current research issues and applications*, Lecture Notes in Computer Science, vol. 1028, Springer, Berlin, Germany, 1996.

[Aye63] F. Ayers, Jr., *Mathematics of finance*, Shaum's Outline, McGraw-Hill, New York, NY, 1963.

[AZ01] R. Alt and H. D. Zimmerman, *Preface: Introduction to special section — business models*, Electronic Markets **11** (2001), no. 1, 3–9.

[Bar01] D. Barkai, *Technologies for sharing and collaborating on the net*, Proceedings of the 1st International Conference on Peer-to-Peer Computing (Lingkoping, Sweden), 2001.

[Bau99] P. Baumard, *Tacit knowledge in organizations*, Sage, London, UK, 1999.

[BCK00] Hemant K. Bhargava, Vidyanand Choudhary, and Ramayya Krishnan, *Pricing and product design: Intermediary strategies in an electronic market*, International Journal of Electronic Commerce **5** (2000), no. 1, 37–56.

[BDBW97] Jeffrey M. Bradshaw, Stewart Dutfield, Pete Benoit, and John D. Woolley, *KAoS: Toward an industrial-strength open agent architecture*, Software agents (Jeffrey M. Bradshaw, ed.), AAAI Press, 1997, pp. 375–418.

[BDD00] Christopher H. Brooks, Edmund H. Durfee, and Rajarshi Das, *Price wars and niche discovery in an information economy*, Proceedings of ACM Conference on Electronic Commerce (EC-00) (Minneapolis, MN), October 2000.

[BDLT00] Roger W.H. Bons, Frank Dignum, Ronald M. Lee, and Yao-Hua Tan, *A formal analysis of auditing principles for electronic trade procedures*, International Journal of Electronic Commerce **5** (2000), no. 1, 57–82.

[Ben88] J. Bennett, *Events and their names*, Oxford University Press, 1988.

[Ben03] John Benyon, *Building police co-operation: The European construction site around the third pillar*, World Wide Web file, Accessed 28 February 2003, Dated 1996.
`http://www.psa.ac.uk/cps/1996/beny.pdf`.

[BH79] Kent Bach and Robert M. Harnish, *Linguistic communication and speech acts*, The MIT Press, Cambridge, Massachusetts, 1979.

[Bha90] Hemant Kumar Bhargava, *A logic model for model management: An embedded languages approach*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1990, Available as a technical report from the Department of Operations and Information Management.

[BI94] Michael Bieber and Thomás Isakowitz, *Text editing and beyond: A study in logic modeling*, Decision Support Systems **11** (1994), no. 2, 219–240.

[Bic01] M. Bichler, *The future of e-markets: Multi-dimensional market mechanisms*, Cambridge University Press, Cambridge, United Kingdom, 2001.

[BK95] Hemant K. Bhargava and Steven O. Kimbrough, *On embedded languages, meta-level reasoning and computer-aided modeling*, The Impact of Emerging Technologies on Computer Science and Operations Research (Stephen G. Nash and Ariela Sofer, eds.), Kluwer Academic Publishers, Boston, MA, 1995, ISBN 0-7923-9542-5. File: csts-94-meta-sok-hkb., pp. 27–44.

[BKM97] Hemant K. Bhargava, Ramayya Krishnan, and Rudolf Müller, *Electronic commerce in decision technologies: A business cycle analysis*, International Journal of Electronic Commerce **1** (1997), no. 4, 109–128.

[BL88] Marvin Belzer and Barry Loewer, *A conditional logic for defeasible beliefs*, Decision Support Systems **4** (1988), no. 1, 129–142.

[BL95] James Baty and Ronald M. Lee, *Intershop: Enhancing the vendor/customer dialectic in electronic shopping*, Journal of Managment Information Systems (1995).

[Bla90] Robert W. Blanning, *The sensitivity properties of hierarchical logic-based models*, Decision Support Systems **6** (1990), no. 2, 89–97.

[Bla94] ———, *A relational algebra for propositional logic*, Decision Support Systems **11** (1994), no. 2, 211–218.

[BLW97] Roger W.H. Bons, Ronald M. Lee, and Renée Wagenaar, *Computer-aided auditing of inter-organizational trade procedures*, Intelligent Systems in Accounting, Finance and Management **6** (1997), no. 2, 29–46.

[BLWW95] Roger W.H. Bons, Ronald M. Lee, Renée W. Wagenaar, and Clive D. Wrigley, *Modelling inter-organizational trade procedures using documentary Petri nets*, Proceedings of the Hawaii International Conference on System Sciences, 1995.

[BM55] R. R. Bush and F. Mosteller, *Stochastic models for learning*, Wiley, New York, NY, 1955.

[BM85] David C. Blair and M. E. Maron, *An evaluation of retrieval effectiveness for a full-text document-retrieval system*, Communications of the ACM **28** (1985), no. 3, 289–299.

[BMS00] B. Banerjee, R. Mukherjee, and S. Sen, *Learning mutual trust*, Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies, 2000, citeseer.nj.nec.com/banerjee00learning.html, pp. 9–14.

[BO03] D. W. Bunn and F. Oliveira, *Evaluating individual market power in electricity markets via agent-based simulation*, Annals of Operations Research **121** (2003), 57–78.

[Boi98] Max Boisot, *Knowledge assets: Securing competitive advantage in the information economy*, Oxford University Press, Oxford, UK, 1998.

[Bon97] Roger W.H. Bons, *Designing trustworthy trade procedures for open electronic commerce*, Ph.D. thesis, EURIDIS at Erasmus University, Rotterdam, The Netherlands, September 1997.

[BRJ99] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language user guide*, Addison-Wesley, 1999.

[BS00] Adam M. Brandenburger and Harborne (Gus) W. Stuart, *Biform games*, Working paper No. 00-06, Harvard NOM Research Paper, http://ssrn.com/abstract=264199, November 2000.

[Buh98] R. A. J. Buhr, *Use case maps as architectural entities for complex systems*, IEEE Transactions on Software Engineering **24** (1998), no. 12, 1131–1155.

[Cam90] Colin F. Camerer, *Behavioral game theory*, Insights in Decision Making: A Tribute to Hillel J. Einhorn (R.M. Hogarth, ed.), Univ. of Chicago Press: Chicago, IL, 1990, pp. 311–336.

[Cam03] ———, *Behavioral game theory: Experiments in strategic interaction*, Russell Sage Foundation and Princeton University Press, New York, NY and Princeton, NJ, 2003.

[Cas82] H. N. Casteñeda, *The logical structure of legal systems: A new perspective*, Deontic Logic, Computational Linguistics and Legal Information Systems (L.E. Allen, ed.), vol. II, North-Holland Publishing Company, 1982, pp. 21–37.

[Cau94] Robert L. Causey, *EVID: A system for interactive defeasible reasoning*, Decision Support Systems **11** (1994), no. 2, 103–131.

[CB98] Caroline Claus and Craig Boutilier, *The dynamics of reinforcement learning in cooperative multiagent systems*, Proceedings of the Fifteenth National Conference on Artificial Intelligence (Menlo Park, CA), AAAI Press/MIT Press, 1998, pp. 746–752.

[CCS97] L. Cholvy, F. Cuppens, and C. Saurel, *Towards a logical formalization of responsibility*, Proceedings of the Sixth International Conference on Artificial Intelligence and the Law (Melbourne, Australia), ACM Press, 1997, pp. 233–242.

[CH98] Colin F. Camerer and Teck-Hua Ho, *Experience-weighted attraction learning in coordination games: Probability rules, heterogeneity, and time-variation*, Journal of Mathematical Psychology **42** (1998), 305–326.

[CH99] ———, *Experience-weighted attraction learning in normal form games*, Econometrica **67** (1999), no. 4, 827–974.

[Che92] K. T. Chen, *Schematic evaluation of internal accounting control systems*, Ph.D. thesis, University of Texas at Austin, Austin, Texas, 1992.

[Che99] Fangrou Chen, *Decentralized supply chains subject to information delays*, Management Science **45** (1999), no. 8, 1076–1090.

[CK73] C.C. Chang and H.J. Keisler, *Model theory*, A. Heyting, et al. (eds.) Studies in Logic and The Foundations of Mathematics, vol. 73, North-Holland and American Elsevier Publishing Co. Inc., Amsterdam and New York, NY, 1973.

[CK97] Ivo Cathomen and Stefan Klein, *The development of FEDI in Switzerland: A life-cycle approach*, International Journal of Electronic Commerce **1** (1997), no. 4, 129–145.

[CK04] Alex K. Chavez and Steven O. Kimbrough, *A model of human behavior in coalition formation games*, Proceedings of the 4th Annual International Conference on Cognitive Modeling, July 2004.

[CL03] K.T. Chen and Ronald M. Lee, *Knowledge-based evaluation of internal accounting control systems – a pattern recognition approach*, Proceedings of American Accounting Association Conference (Honolulu, Hawaii), 2003.

[CMP90] Philip R. Cohen, Jerry Morgan, and Martha E. Pollack (eds.), *Intentions in communication*, System Development Foundation Benchmark, The MIT Press, Cambridge, Massachusetts, 1990.

[Col95] Andrew M. Colman, *Game theory and its applications in the social and biological sciences*, second ed., Routledge, London, UK, 1995.

[Cou97] A. Cournot, *Researches into the mathematical principles of the theory of wealth*, Macmillan, New York, NY, 1897, English edition edited by N. Bacon. Originally published in French as *Recherches sur Principes Mathématiques de la Théorie des Richesses* in 1838.

[Cov97] Michael A. Covington, *On designing a language for electronic commerce*, International Journal of Electronic Commerce **1** (1997), no. 4, 31–48.

[Cov98] ———, *Speech acts, electronic commerce, and KQML*, Decision Support Systems **22** (1998), no. 3, 203–211.

[Cre73] M.J. Cresswell, *Logics and languages*, Nethuen & Co. Ltd., London, UK, 1973.

[CSLC01] S. Choi, P. M. Samuel, J. Liu, and S. P. Chan, *A genetic agent-based negotiation system*, Computer Networks **37** (2001), no. 2, 195–204.

[D+01] N. Damianou et al., *The Ponder policy specification language*, Proceedings of the International Workshop POLICY 2001 (Berlin, Germany) (M. Sloman, ed.), Springer Verlag, LNCS, 2001.

[Das99] Aspassia Daskalopulu, *Logic-based tools for the analysis and representation of legal contracts*, Ph.D. thesis, Department of Computing, Imperial College, University of London, 1999.

[Dav80] D. Davidson, *Essays on actions and events*, Clarendon Press, 1980.

[Dav96] T. Davis, *Lexical semantics and linking in the hierarchical lexicon*, Ph.D. thesis, Stanford University, 1996.

[Daw80] Robyn M. Dawes, *Social dilemmas*, Annual Review of Psychology **31** (1980), 169–193.

[Daw82] Richard Dawkins, *The extended phenotype : The gene as the unit of selection*, Oxford University Press, Oxford, UK, 1982.

[Daw89] ———, *The selfish gene*, Oxford University Press, New York, NY, 1989.

[DB01] T. Dimitrakos and J. Bicarregui, *Towards a framework for managing trust in e-services*, Proceedings of the Fourth International Conference on Electronic Commerce Research, IFIP INFORMS, 2001, pp. 360–381.

[DDLS01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, *The Ponder policy specification language*, Lecture Notes in Computer Science **1995** (2001), 18–38.

[DDM01] A. Daskalopulu, T. Dimtrakos, and T.S.E. Maibaum, *E-contract fulfillment and agents' attitudes*, Proceedings ERCIM WG E-Commerce Workshop on the Role of Trust in E-Business (Zurich), October 2001.

[DDM02] Aspassia Daskalopulu, Theo Dimitrakos, and Tom Maibaum, *Evidence-based electronic contract performance monitoring*, Group Decision and Negotiation **11** (2002), no. 6, 469–485.

[Dek03] H. Dekker, *Control of inter-organizational relationships: Evidence on appropriation concerns and coordination requirements*, Accounting, Organizations and Society **29** (2003), 27–49.

[Del03] Chrysanthos N. Dellarocas, *Efficiency and robustness of binary feedback mechanisms in trading environments with moral hazard*, Working Paper No. 4297-03, MIT, Sloan School, Cambridge, MA, January 2003, http://ssrn.com/abstract=393043.

[Dew92] S. D. Dewitz, *Contracting on a performative network: Using information technology as a legal intermediary*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[DG00] Frank Dignum and Mark Greaves (eds.), *Issues in agent communication*, Lecture notes in computer science, vol. 1916, Springer, 2000.

[Dic00] Kevin Dick, *XML: A manager's guide*, Addison-Wesley, Reading, Massachusetts, 2000, ISBN: 0-201-43335-4.

[dJ01] Aldo de Moor and Manfred A. Jeusfeld, *Making workflow change acceptable*, Requirements Engineering **6** (2001), no. 4, 75–96.

[DK99] Frank Dignum and Ruurd Kuiper, *Specifying deadlines with continuous time using deontic and temporal logic*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 67–86.

[DKL96] Garett O. Dworman, Steven O. Kimbrough, and James D. Laing, *On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game*, Journal of Management Information Systems **12** (1995–96), no. 3, 97–125.

[DKL95a] ———, *Bargaining in a three-agent coalitions game: An application of genetic programming*, Working Notes: AAAI–95 Fall Symposium Series, Genetic Programming (Boston, MA, November 10-12, 1995), AAAI, 1995, pp. 9–16.

[DKL95b] ———, *On automated discovery of models using genetic programming in game–theoretic contexts*, Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences, Volume III: Information Systems: Decision Support and Knowledge-Based Systems (Los Alamitos, CA) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1995, pp. 428–438.

[DKL96] ———, *Bargaining by artificial agents in two coalition games: A study in genetic programming for electronic commerce*, Genetic Programming 1996: Proceedings of the First Annual Genetic Programming Conference, July 28-31, 1996, Stanford University (John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, eds.), The MIT Press, 1996, pp. 54–62.

[DL89] Sandra D.Dewitz and Ronald M. Lee, *Legal procedures as formal conversations: Contracting on a performative network*, Proceedings of the Tenth International Conference on Information Systems (Baltimore, Maryland) (Janice I. DeGross, John C. Henderson, and Benn R. Konsynski, eds.), Association for Computing Machinery, December 4-6, 1989, pp. 53–65.

[Don66] Keith S. Donnellan, *Reference and definite descriptions*, The Philosophical Review **75** (1966), no. 3, 281–304.

[Dow78] D.R. Dowty, *A guide to Montague's PTQ*, Indiana University Linguistics Club, 1978.

[Dri97] Chris Driscoll, *XML touted as cure for EDI ills: New markup language extends web capabilities beyond HTML's limits*, Web page, 5 August 1997, `www.geocities.com/WallStreet/Floor/5815/edinews01.htm`, accessed 1999-12-28.

[DRL94] Sandra K. Dewitz, Young Ryu, and Ronald M. Lee, *Defeasible reasoning in law*, Decision Support Systems **11** (1994), no. 2, 133–155, Typographical error in publication. The first author is Sandra D. Dewitz.

[DS97] Aspassia Daskalopulu and Marek Sergot, *The representation of legal contracts*, AI and Society **11** (1997), no. 1, 6–17.

[DS02] ———, *Computational aspects of the FLBC framework*, Decision Support Systems **33** (2002), no. 3, 267–290.

[DT98] T. K. Das and B. S. Teng, *Between trust and control: Developing confidence in partner cooperation in alliances*, Academy of Management Review **23** (1998), no. 3, 491–512.

[dv96] B. de Wit and M. van Delden, *Performance analysis of internal communication: A research approach for assessing the quality of policy and motivating communication*, The Quality of Communication in Organisations in Theory and Practice (J. A. de Ridder and K. Seisveld, eds.), Cramwinckel, Amsterdam, The Netherlands, 1996, Translated from Dutch.

[DV98] J. Dietz and V. Van Reijswoud, *DEMO modelling handbook*, Technical report, 1998, Volume 2.

[dv01] Aldo de Moor and W. J. van den Heuvel, *Making virtual communities work: Matching their functionalities*, Proceedings of the 9th International Conference on Conceptual Structures (Palo Alto, CA), Stanford University, July 30–August 3, 2001.

[Emm93] Margaret A. Emmelhainz, *EDI: A total management guide*, second ed., Van Nostrand Reinhold, New York, NY, 1993, ISBN: 0-442-312690-9.

[Emm94] ———, *Electronic data interchange in logistics*, The Logistics Handbook (James F. Robeson and William C. Copacino, eds.), The Free Press, New York, NY, 1994, ISBN: 0-02-926595-9., pp. 737–756.

[ER98] Ido Erev and Alvin E. Roth, *Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria*, The American Economic Review **88** (1998), no. 4, 848–881.

[Fan03] Christina Fang, *Strategy as valuation*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 2003.

[Far98] P. Faratin, *Negotiation among groups of autonomous computational agents. thesis*, Ph.D. thesis, Department of Electronic Engineering, Queen Mary and Westfield College, University of London, 1998.

[FBH86] Eileen Fitzpatrick, Joan Bachenko, and Don Hindle, *The status of telegraphic sublanguages*, Analyzing Language in Restricted Domains: Sublanguage Description and Processing (Ralph Grishman and Richard Kittredge, eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1986, pp. 39–51.

[Fel98] Christiane Fellbaum (ed.), *Wordnet: An electronic lexical database*, The MIT Press, Cambridge, MA, 1998, ISBN: 0-262-06197-X.

[FFMM94a] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire, *KQML as an agent communication language*, Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94), ACM Press, November 1994.

[FFMM94b] ———, *KQML as an agent communication language*, The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94), ACM, ACM Press, November 1994.

[FJ97] Sergio Focardi and Caroline Jonas, *Modeling the market: New theories and techniques*, Frank J. Fabozzi Associates, New Hope, Pennsylvania, 1997.

[FKP+02] Christina Fang, Steven O. Kimbrough, Stefano Pace, Annapurna Valluri, and Zhiqiang Zheng, *On adaptive emergence of trust behavior in the game of stag hunt*, Group Decision and Negotiation **11** (November 2002), no. 6, 449–467.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke, *The anatomy of the grid: Enabling scalable virtual organizations*, International Journal of Supercomputer Applications and High Performance Computing **15** (2001), no. 3.

[FL81] C. F. Flores and J. Ludlow, *Doing and speaking in the office*, DSS: Issues and Challenges (G. Fick and R. Sprague, eds.), Pergamon Press, London, UK, 1981.

[FL98] Drew Fudenberg and David K. Levine, *The theory of learning in games*, The MIT Press, Cambridge, MA, 1998.

[FM86] D. Fundenberg and E. Maskin, *The folk theorem with discounting and with incomplete information*, Econometrica **54** (1986), 533–554.

[Fod86] J.A. Fodor, *Meaning, convention, and the blue book*, Meaning (J.V. Canfield, ed.), Garland Publishing, New York, NY, 1986, pp. 87–108.

[Fos82] Antony Charles Foskett, *The subject approach to information*, 4th ed., Linnet Books, Hamden, CT, 1982.

[Fou97] Foundation for Intelligent Physical Agents, *FIPA 97 specification part 2: Agent communication language*, November 28, 1997, Geneva, Switzerland.

[Fre93] Gottlob Frege, *Grundgesetze der arithmetik*, Verlag Hermann Pohle, Jena, Germany, 1893, Partial translation as *The Basic Laws of Arithmetic* by M. Furth, Berkeley: U. of California Press, 1964.

[Fre04] Free Software Foundation, Inc., *GNU CLISP — an ANSI Common Lisp*, http://clisp.cons.org/, June 2004.

[Fri77] J. W. Friedman, *Oligopoly and the theory of games*, North Holland (now Elsevier), 1977.

[FSW00] Ming Fan, Jan Stallaert, and Andrew B. Whinston, *The internet and the future of financial markets*, Communications of the ACM **43** (2000), no. 11, 82–88.

[FW+93] Tim Finin, Jay Weber, et al., *Draft specification of the KQML agent–communication language plus example agent policies and architectures*, Manuscript obtained from `http://www.cs.umbc.edu`, 1993.

[GA03] J. Gordijn and J. M. Akkermans, *Value based requirements engineering: Exploring innovative e-commerce ideas*, Requirements Engineering Journal **8** (2003), no. 2, 114–134.

[Gal90] Charles R. Gallistel, *The organization of learning*, The MIT Press, Cambridge, MA, 1990.

[GHB99] Mark Greaves, Heather Holback, and Jeffrey Bradshaw, *What is a conversation policy?*, Proceedings for the Workshop on Specifying and Implementing Conversation Policies (Seattle, WA) (Mark Greaves

and Jeffrey Bradshaw, eds.), Autonomous Agents '99, May 1, 1999, pp. 1–9.

[GKS97] Georg Geyer, Christoph Kuhn, and Beat Schmid, *Designing a market for quantitative knowledge*, International Journal of Electronic Commerce **1** (1997), no. 4, 89–108.

[GLC99] B.N. Grosof, Y. Labrou, and N.Y. Chan, *A declarative approach to business rules in contracts: Courteous logic programs in XML*, Proceedings 1st ACM Conference on Electronic Commerce (EC-99) (M.P. Wellman, ed.), November 1999.

[GM99] Guido L. Geerts and William E. McCarthy, *An accounting object infrastructure for knowledge-based enterprise models*, IEEE Intelligent Systems **14** (1999), no. 4, 89–94.

[GMS98] Patrick Grim, Gary Mar, and Paul St. Denis, *The philosophical computer: Exploratory essays in philosophical computer modeling*, The MIT Press, Cambridge, MA, 1998.

[Gol89] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley: Reading, MA, 1989.

[Gor02] J. Gordijn, *Value-based requirements engineering: Exploring innovative e-commerce ideas*, Ph.D. thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2002, `http://www.cs.vu.nl/~gordijn`.

[GP98] Charles F. Goldfarb and Paul Prescod, *The xml handbook*, Prentice Hall PTR, Upper Saddle River, NJ, 1998, ISBN: 0-13-081152-1.

[GPW98] J. S. Gans, D. Price, and K. Woods, *Contracts and electricity pool prices*, Australian Journal of Management **23** (1998), no. 1, 83–96.

[Gri57] Paul Grice, *Studies in the way of words*, ch. Meaning, pp. 213–223, Harvard University Press, Cambridge, MA, 1989 (originally 1957), .

[GS93] Dhananjay K. Gode and Shyam Sunder, *Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality*, Journal of Political Economy **101** (1993), no. 1, 119–137.

[GVN94] P. Geerts, D. Vermeir, and D. Nute, *Ordered logic: Defeasible reasoning for multiple agents*, Decision Support Systems **11** (1994), no. 2, 157–190.

[Har68] Zellig Harris, *Mathematical structures of language*, John Wiley & Sons, New York, NY, 1968.

[Har87] David Harel, *Statecharts: A visual formalism for complex systems*, Science of Computer Programming **8** (1987), 231–274.

[Har02] Richard K. Harrison, *Bibliography of planned languages (excluding Esperanto)*, World Wide Web, 1992-2002, accessed December 2002, `www.invisiblelighthouse.com/langlab/bibliography.html`.

[HBC97] E.N. Hanson, S. Bodagala, and U. Chadaga, *Optimized trigger condition testing in Ariel using Gator networks*, Tech. Report UF-CIS-TR-97-021, CISE Department, University of Florida. Gainesville, FL (USA), November 1997.

[Her96] Henning Herrestad, *Formal theories of rights*, Ph.D. thesis, University of Oslo, Oslo, Norway, 1996, Available as ISBN 82-7833-015-0, Karnovs Forlag, publisher.

[HH00] S. M. Harvey and W. W. Hogan, *California electricity prices and forward market hedging*, Technical report: working paper series, Center

for Business and Government, John F. Kennedy School of Government, Harvard University, Cambridge, Massachusetts 02138, October 2000.

[HK98] Gary H. Hua and Steven O. Kimbrough, *On hypermedia-based argumentation decision support systems*, Decision Support Systems **22** (1998), no. 3, 259–275.

[Hoh78] W.N. Hohfeld, *Fundamental legal conceptions as applied in judicial reasoning*, Greenwood Press Publishers, 1978.

[Hol85] Charles A. Holt, *An experimental test of the consistent-conjectures hypothesis*, The American Economic Review **75** (1985), no. 3, 314–325.

[Hol92] John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, 1992.

[Hoo88] J. N. Hooker, *A quantitative approach to logical inference*, Decision Support Systems **4** (1988), no. 1, 45–69.

[Hoo02] Christopher Hookway, *Truth, rationality, and pragmatism: Themes from peirce*, Clarendon Press, Oxford, UK, 2002.

[HP81] J. Michael Harrison and Stanley R. Pliska, *Martingales and stochastic integrals in the theory of continuous time trading*, Stochastic Processes and their Applications **11** (1981), 215–260.

[HS98] Michael N. Huhns and Munindar P. Singh (eds.), *Readings in agents*, Morgan Kaufmann, San Francisco, CA, 1998, ISBN: 1-55860-495-2.

[Hua95] Hua Hua, *Theory and applications of argumentation support systems*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1995, Available as a technical report from the Department of Operations and Information Management.

[Hum78] David Hume, *A treatise of human nature*, Oxford: Clarendon, Oxford, UK, [1739] 1978, Selby-Bigge, ed.

[HV99] Tamer M. Özsu and P. Valduriez, *Principles of distributed database systems*, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[HW98] J. Hu and M. P. Wellman, *Multiagent reinforcement learning: Theoretical framework and an algorithm*, Fifteenth International Conference on Machine Learning, July 1998, pp. 242–250.

[IBM04] IBM, *IBM Optimization Solutions and Library*, 2004, http://www-3.ibm.com/software/data/bi/osl/index.html.

[ICC94] ICC, *The uniform customs and practices for documentary credit procedures*, International Chamber of Commerce publication 500, January 1994, Paris, France.

[IU99a] Kiyoshi Izumi and Kazuhiro Ueda, *Analysis of dealers' processing financial news based on an artificial market approach*, Journal of Computational Intelligence in Finance **7** (1999), 23–33.

[IU99b] ———, *Analysis of exchange rate scenarios using an artificial market approach*, Proceedings of the International Conference on Artificial Intelligence (A. Amin, C.-H. Chen, and et. al, eds.), CSREA Press, 1999, pp. 360–366.

[J+93] Edward Johnson et al., *Policespeak: Police communications and language, English-French lexicon*, PoliceSpeak Publications, Cambridge Research Laboratories, 181a Huntingdon Road, Cambridge, CB3 ODJ, 1993, ISBN: 1 898211 01 9.

[Jac02] Ray Jackendoff, *Foundations of language: Brain, meaning, grammar, evolution*, Oxford University Press, Oxford, UK, 2002.

[Jer88] Robert G. Jeroslow, *Spatial imbedding for linear and for logic structures*, Decision Support Systems **4** (1988), no. 1, 71–86.

[JK04] Andrew J.I. Jones and Steven O. Kimbrough, *A note on modeling speech acts as signalling conventions*, Formal Modelling for Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, Berlin, Germany, 2004.

[JM00] D.S. Jurafsky and J.H. Martin, *Speech and language processing*, Prentice Hall, 2000.

[Joh98] Edward Johnson, *LinguaNet final report*, World Wide Web file, 1998, http://www.hltcentral.org/usr_docs/project-source/linguanet/Final-Report/Final-Report.html, accessed 28 February 2003.

[Joh02] ———, *Talking across frontiers*, Proceedings of the International Conference on European Cross-Border Co-operation, 2002, Available at: http://www.prolingua.co.uk/talking.pdf. Accessed December 2002.

[Jon02] Andrew J.I. Jones, *On the concept of trust*, Decision Support Systems **33** (2002), no. 3, 225–232.

[Jon04] ———, *A logical framework*, Open Agent Societies: Normative Specifications in Multi-Agent Systems (Jeremy Pitt, ed.), John Wiley & Sons, Chichester, UK, 30 December 2004, ISBN: 047148668X.

[JP04] Andrew J.I. Jones and X. Parent, *Conventional signalling acts and conversation*, Advances in Agent Communication (Frank Dignum, ed.), Lecture Notes in Artificial Intelligence, Volumn 2922, Springer-Verlag, Berlin, Heidelberg, New York, 2004, pp. 1–17.

[JPB99] S. Joshi, J. Parker, and M. A. Bedau, *Technical trading creates prisoner's dilemma: Results from an agent-based model*, Computational Finance (Cambidge, MA) (Y. S. Abu-Mostafa, Blake LeBaron, A. W. Lo, and A. S. Weigend, eds.), The MIT Press, 1999.

[JS96] Andrew J.I. Jones and Marek J. Sergot, *A formal characterisation of institutionalised power*, Journal of the Interest Group in Pure and Applied Logic (IGPL) **4** (1996), no. 3, 427–443, Reprinted in [V$^+$97, pages 349–367].

[KA88] Steven O. Kimbrough and Fred Adams, *Why nonmonotonic logic?*, Decision Support Systems **4** (1988), 111–127.

[KHG00] Jeffry O. Kephart, James E. Hansen, and Amy R. Greenwald, *Dynamic pricing by software agents*, Computer Networks **32** (2000), no. 6, 731–752.

[KHL$^+$98a] Jeffrey O. Kephart, James E. Hanson, David W. Levine, Benjamin N. Grosof, Jakka Sairamesh, Richard B. Segaland, and Steve R. White, *Dynamics of an information-filtering economy*, Proceedings of the Second International Workshop on Cooperative Information Agents, July 1998.

[KHL$^+$98b] Jeffry O. Kephart, James E. Hansen, D. W. Levine, Benjamin Grosof, J. Sairamesh, R. Segal, and S. R. White, *Emergent behavior in information economies*, Proceedings of the International Conference on Multi-Agent Systems, 1998.

[Kim99] Steven O. Kimbrough, *Formal language for business communication: Sketch of a basic theory*, International Journal of Electronic Commerce **3** (Winter 1998–99), no. 2, 23–44.

[Kim80] ———, *The concepts of fitness and selection in evolutionary biology*, Journal of Social and Biological Structures **3** (1980), 149–170.

[Kim90] ———, *On representation schemes for promising electronically*, Decision Support Systems **6** (1990), no. 2, 99–121.

[Kim91] Paul Kimberley, *Electronic data interchange*, McGraw-Hill, Inc., New York, New York, 1991.

[Kim97] Steven O. Kimbrough, *On electronic commerce, subatomic semantics and Cæsar's stabbing*, Proceedings of the Thirtieth Hawaii International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1997, pp. 361–370.

[Kim98a] ———, *On ESΘ theory and the logic of the X12 date/time qualifiers*, Proceedings of the Thirty-First Hawai'i International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1998, pp. 330–339.

[Kim98b] ———, *Sketch of a basic theory for a formal language for business communication*, Proceedings of the Thirty-First Hawai'i International Conference on System Sciences (Los Alamitos. CA) (Ralph H. Sprague, Jr., ed.), IEEE Press, 1998, pp. 717–725.

[Kim01] ———, *Reasoning about the objects of attitudes and operators: Towards a disquotation theory for representation of propositional content*, Proceedings of ICAIL '01, International Conference on Artificial Intelligence and Law, 2001.

[Kim02] ———, *A note on the Good Samaritan paradox and the disquotation theory of propositional content*, Proceedings of ΔEON'02, Sixth International Workshop on Deontic Logic in Computer Science (John Horty and Andrew J.I. Jones, eds.), May 2002, pp. 139–148.

[Kim03] ———, *Computational modeling and explanation: Opportunities for the information and management sciences*, Computational Modeling and Problem Solving in the Networked World: Interfaces in Computing and Optimization (Hemant K. Bhargava and Nong Ye, eds.), Operations Research/Computer Science Interfaces Series, Kluwer, Boston, MA, 2003, pp. 31–57.

[Kin01] Jeffrey C. King, *Complex demonstratives: A quantificational account*, MIT Press, Cambridge, MA, 2001.

[KK01] Kyra Karmiloff and Annette Karmiloff-Smith, *Pathways to language: From fetus to adolescent*, Harvard University Press, Cambridge, MA, 2001.

[KL86] Steven O. Kimbrough and Ronald M. Lee, *On illocutionary logic as a telecommunications language*, Proceedings of the International Conference on Information Systems (San Diego, CA), December 1986, pp. 15–25.

[KL88] ———, *Logic modeling: A tool for management science*, Decision Support Systems **4** (1988), no. 1, 3–16.

[KL97] ———, *Formal aspects of electronic commerce: Research issues and challenges*, International Journal of Electronic Commerce **1** (1997), no. 4, 11–30.

[KL03] Steven O. Kimbrough and Ming Lu, *A note on Q-learning in the Cournot game*, WeB 2003: Proceedings of the Second Workshop in e-Business (Seattle, WA), December 13-14, 2003, Available at http://opim-sun.wharton.upenn.edu/~sok/sokpapers/2004/cournot-rl-note-final.doc.

[KL04] ———, *Simple reinforcement learning agents: Pareto beats Nash in an algorithmic game theory study*, Information Systems and e-Business (forthcoming 2004).

[KLK04] Steven O. Kimbrough, Ming Lu, and Ann Kuo, *A note on strategic learning in policy space*, Formal Modelling in Electronic Commerce: Representation, Inference, and Strategic Interaction (Steven O. Kimbrough and D. J. Wu, eds.), Springer, 2004.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KLN84] Steven O. Kimbrough, Ronald M. Lee, and David N. Ness, *Performative, informative and emotive systems: The first piece of the PIE*, Proceedings of the Fifth International Conference on Information Systems (Tucson, AZ) (Leslie Maggie et al., ed.), November 28-30, 1984, pp. 141–148.

[KLPY03] Steven O. Kimbrough, Thomas Y. Lee, Balaji Padmanabhan, and Yinghui Yang, *On original generation of structure in legal documents*, Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL), 2003.

[KM93a] Steven O. Kimbrough and Scott A. Moore, *On obligation, time, and defeasibility in systems for electronic commerce*, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III, Information Systems: DSS/Knowledge-Based Systems (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, 1993, pp. 493–502.

[KM93b] ———, *On the spanning hypothesis for EDI semantics*, Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences (Los Alamitos, California) (Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., eds.), IEEE Computer Society Press, January 1993.

[KM97] ———, *On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity*, ACM Transactions on Information Systems **15** (October 1997), no. 4, 321–367.

[KMM80] D. Kalish, R. Montague, and G. Mar, *Logic: Techniques of formal reasoning*, second edition ed., Harcourt Brace Jovanovich, Inc., New York, 1980.

[Koz00] John R. Koza, *Genetic programming: On the programming of computers by means of natural selection*, A Bradford Book/The MIT Press, 1992 (7th printing, 2000), ISBN 0-262-11170-5.

[KPL03] Gary Kleinman, Dan Palmon, and Picheng Lee, *The effects of personal and group level factors on the outcomes of simulated auditor and client teams*, Group Decision and Negotiation **11** (2003), no. 1, 57–84.

[KR74] James P. Kahan and Amnon Rapoport, *Test of the bargaining set and kernel models in three-person games*, Game Theory as a Theory of Conflict Resolution (Anatol Rapoport, ed.), D. Reidel, Dordrecht, The Netherlands, 1974, pp. 119–160.

[KR84] ———, *Theories of coalition formation*, Lawrence Earlbaum Associates, Hillsdale, NJ, 1984.

[KR95] John H. Kagel and Alvin E. Roth (eds.), *The handbook of experimental economics*, Princeton University Press, Princeton, NJ, 1995.

[Kre90] David M. Kreps, *Game theory and economic modeling*, Clarendon Press, Oxford, England, 1990.

[Kri63] S. Kripke, *Semantical considerations on modal logics*, Acta Philosophica Fennica, Modal and Many-Valued Logics (1963), 83–94.

[Kri90] Ramayya Krishnan, *A logic modeling language for automated model construction*, Decision Support Systems **6** (1990), no. 2, 123–152.

[Kro97] Cristen Krogh, *Normative structures in natural and artificial systems*, Ph.D. thesis, University of Oslo, Oslo, Norway, 1997, Available at http://www.uio.no/∼krogh/ documents/papers/complex97.ps.

[KS98] Ioanis Karatzas and Steven Shreve, *Methods of mathematical finance*, Springer-Verlag, new York, New York, 1998.

[KS03] D.C.L. Kuo and M. Smits, *Performance of integrated supply chains: An international case study in high tech manufacturing*, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36) (Ralph H. Sprague, Jr., ed.), IEEE Computer Society Press, Los Alamitos, CA, January 2003, (CD-ROM).

[KT00] Steven O. Kimbrough and Yao-Hua Tan, *On lean messaging with unfolding and unwrapping for electronic commerce*, International Journal of Electronic Commerce **5** (2000), no. 1, 83–108.

[KWZ02] Steven O. Kimbrough, D. J. Wu, and Fang Zhong, *Computers play the beer game: Can artificial agents manage supply chains?*, Decision Support Systems **33** (2002), no. 3, 323–333.

[KY04] Steven O. Kimbrough and Yinghui Yang, *Action at the tables: Sketching a tabular representation for utterances under the language–action perspective*, Proceedings of the 9th International Working Conference on the Language Action Perspective on Communication Modelling (Rutgers, The State University of New Jersey, New Brunswick, NJ) (Mark Aakhus and Mikael Lind, eds.), School of Communication, Information, and Library Studies, 2–3 June 2004, http://www.scils.rutgers.edu/˜aakhus/lap/lap04.htm, pp. 103–120.

[LA87] Diane J. Litman and James F. Allen, *A plan recognition model for subdialogues in conversations*, Cognitive Science **11** (1987), 163–200.

[Lam91] Karel Lambert (ed.), *Philosophical applications of free logic*, Oxford University Press, Oxford, U.K., 1991.

[LAP99] Blake LeBaron, W. B. Arthur, and R. G. Palmer, *The time series properties of an artificial stock market*, Journal of Economic Dynamics and Control **23** (1999), 1487–1516.

[LAP04] LAP, *Home LAP '04*, World Wide Web page, September 2004, `http://www.scils.rutgers.edu/∼aakhus/lap/lap04.htm`. LAP=Language/Action Perspective.

[Lar85] Andrew Large, *The artificial language movement*, Basil Blackwell, New York, NY, 1985, ISBN 0-631-14497-8.

[LB96] Ronald M. Lee and Roger W.H. Bons, *Soft-coded trade procedures for open-edi*, International Journal of Electronic Commerce **1** (1996), no. 1, 27–49.

[LBW01] Ronald M. Lee, Roger W.H. Bons, and René W. Wagenaar., *Pattern-directed auditing of inter-organisational trade procedures*, Proceedings of the 1st IFIP Conference on eCommerce, eBusiness, and eGovernment (Zurich, Switzerland), 4–5 October 2001.

[LeB00] Blake LeBaron, *Agent-based computational finanace: Suggested readings and early research*, Journal of Economic Dynamics and Control **24** (2000), 679–702.

[LeB01a] ———, *A builder's guide to agent-based financial markets*, Quantitative Finance **1** (2001), 254–261.

[LeB01b] ———, *Evolution and time horizons in an agent based stock market*, Macroeconomic Dynamics **5** (2001), 254–261.

[Lee80] Ronald M. Lee, *CANDID: a logical calculus for describing financial contracts*, Ph.D. thesis, The Wharton School, University of Pennsylvania, Philadelphia, PA, 1980, Available as WP-80-06-02, Department of Operations and Information Management (née Decision Sciences).

[Lee88a] ———, *Bureaucracies as deontic systems*, ACM Transactions on Office Information Systems **6** (1988), no. 2, 87–108, Special Issue on the Language/Action Perspective.

[Lee88b] ———, *A logic model for electronic contracting*, Decision Support Systems **4** (1988), no. 1, 27–44.

[Lee92] ———, *Dynamic modeling of documentary procedures: A CASE for EDI*, Proceedings of Third International Working Conference on Dynamic Modeling of Information Systems (Noordwijkerhout, The Netherlands) (H. G. Sol, ed.), June 1992, pp. 95–123.

[Lee98] ———, *INTERPROCS: A Java-based prototyping environment for distributed electronic trade procedures*, Proceedings of the Hawaii International Conference on System Sciences, January 1998, pp. 202–209.

[Lee99] ———, *Distributed electronic trade scenarios: Representation, design, prototyping*, International Journal of Electronic Commerce **3** (1999), no. 2, 105–136.

[Lee02] ———, *Automated generation of electronic procedures: Procedure constraint grammars*, Decision Support Systems **33** (2002), no. 3, 291–308.

[Leh96] Fritz Lehmann, *Machine-negotiated, ontology-based EDI (electronic data interchange)*, Electronic Commerce: Current Research Issues and Applications (Nabil R. Adam and Yelena Yesha, eds.), Lecture Notes in Computer Science, vol. 1028, Springer, Berlin, Germany, 1996, pp. 27–46.

[Lev83] Stephen C. Levinson, *Pragmatics*, Cambridge University Press, Cambridge, England, 1983.

[LF94] Yannis Labrou and Tim Finin, *A semantics approach for KQML—a general purpose communication language for software agents*, Third International Conference on Information and Knowledge Management (CIKM '94), November 1994.

[LF97] ———, *A proposal for a new KQML specification*, Downloaded from `http://www.cs.umbc.edu/kqml/` in January 1998 (Technical Report CS-97-03), February 3, 1997.

[Lig97] Richard Light, *Presenting xml*, SAMS Net, Indianapolis, IN, 1997, ISBN: 1-57521-334-6.

[Lin77] L. Lindahl, *Position and change - a study in law and logic*, Synthese Library, vol. 112, D. Reidel, 1977.

[Lin03] LinguaNet, *The linguanet project*, 2003, `http://www.cbs.dk/-departments/fir/linguanet/`, accessed 2003-02-28.

[LL86] Erkki Lehtinen and Kalle Lyytinen, *Action based model of information systems*, Information Systems **11** (1986), no. 4, 299–317.

[LO02] C. Le Coq and Henrik Orzen, *Do forward markets enhance competition? experimental evidence*, Technical report: working paper series, The Economic Research Institute, Stockholm School of Economics, SSE/EFI Working Paper, Department of Economics, Sveavagen, P.O. Box 6501, 113 83 Stockholm, Sweden, August 2002.

[Lom99] Alessio R. Lomuscio, *Information sharing among ideal agents*, Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, UK, February 1999.

[LR57] R. Duncan Luce and Howard Raiffa, *Games and decisions*, John Wiley, New York, NY, 1957, Reprinted by Dover Books, 1989.

[LS99] Markus A. Lindemann and Beat F. Schmid, *Framework for specifying, building, and operating electronic markets*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 7–22.

[LS89] G. Luger and W. Stubblefield, *Artificial intelligence and the design of expert systems*, Benjamin-Cummings, Redwood City, CA, 1989.

[LS95] Richard Larson and Gabriel Segal, *Knowledge of meaning: An introduction to semantic theory*, The MIT Press, Cambridge, Massachusetts, 1995, ISBN: 0-262-62100-2.

[LS03] H. Ludwig and M. Stolze, *Simple Obligation and Right Model (SORM) for the runtime management of electronic service contracts*, Proceedings of the CAISE–WES, 2003.

[LSDN01] K. Liu, L. Sun, A. Dix, and M. Narasipuram, *Norm-based agency for designing collaborative information systems*, Information Systems Journal **11** (2001), no. 3, 229–247.

[Luo01] Y. Luo, *Contract, cooperation, and performance in international joint ventures*, Strategic Management Journal **23** (2001), 903–919.

[LW86] Ronald M. Lee and George Widmeyer, *Shopping in the electronic marketplace*, Journal of Management Information Systems **2** (1986), no. 4, 21–35.

[LWJ03] Alessio R. Lomuscio, Michael Wooldridge, and Nicholas R. Jennings, *A classification scheme for negotiation in electronic commerce*, Group Decision and Negotiation **11** (2003), no. 1, 31–56.

[Lym96] P. Lyman, *How is the medium the message? notes on the design of networked communication*, Computer Networking and Scholarly Communication in the Twenty-First Century University (T. Stephen, ed.), State University of New York Press, New York, NY, 1996, pp. 39–52.

[Mac30] T. C. Macaulay, *Interlanguage*, The Clarendon Press, Oxford, UK, 1930.

[Mak86] D. Makinson, *On the formal representation of rights relations*, Journal of Philosophical Logic **15** (1986), 403–425.

[McC63] S. McCauley, *Non-contractual relations in business: A preliminary study*, American Sociological Review **28** (1963), 55–67.

[MCC98] D. H. McKnight, L. L. Cummings, and N. L. Chervany, *Initial trust formation in new organizational relationships*, Academy of Management Review **20** (1998), no. 3, 472–490.

[McD85] David McDonald, *Conversations between programs*, Cognitive Constraints on Comunication (Lucia Vaina and Jaakko Hintikka, eds.), Synthese Language Library, vol. 18, D. Reidel Publishing Company, Boston, MA, 1985, ISBN: 90-277-1456-8., pp. 403–424.

[MD04] Scott A. Moore and Kurt Demaagd, *Beer game genetic program*, http://sourceforge.net/projects/beergame/, 2004.

[MF02] Michael W. Macy and Andreas Flache, *Learning dynamics in social dilemmas*, Proceedings of the National Academy of Science (PNAS) **99** (2002), no. suppl. 3, 7229–7236.

[MGM99] P. Maes, R. Guttman, and A. Moukas, *Agents that buy and sell*, Communications of tha ACM **42** (1999), no. 3, 81–91.

[MLF96] James Mayfield, Yannis Labrou, and Tim Finin, *Evaluation of KQML as an agent communication language*, Intelligent Agents Volume II – Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (Berlin, Germany) (M. Wooldridge, J. P. Muller, and M. Tambe, eds.), Springer-Verlag, 1996.

[MMWFF93] Raul Medina-Mora, Terry Winograd, Rofrigo Flores, and Fernando Flores, *The action workflow approach to workflow management technology*, The Information Society **9** (1993), no. 4, 391–404.

[Mon02] Richard Montague, *The proper treatment of quantification in ordinary English*, Formal Semantics: The Essential Readings (Barbara H. Partee Paul Portner, ed.), Blackwell Publishers, 2002, pp. 17–34.

[Moo93] Scott A. Moore, *Saying and doing: Uses of formal languages in the conduct of business*, Ph.D. thesis, University of Pennsylvania, The Wharton School, Philadelphia, PA, 19104, USA, December 1993.

[Moo98] ———, *Categorizing automated messages*, Decision Support Systems **22** (1998), no. 3, 213–241.

[Moo00a] ———, *KQML and FLBC: Contrasting agent communication languages*, International Journal of Electronic Commerce **5** (2000), no. 1, 109–124.

[Moo00b] ———, *On conversation policies and the need for exceptions*, Issues in Agent Communication (Frank Dignum and Mark Greaves, eds.), Lecture Notes in Artificial Intelligence, vol. 1916, Springer, 2000, pp. 144–159.

[Moo01] ———, *A foundation for flexible automated electronic commerce*, Information Systems Research **12** (2001), no. 1, 34–62.

[MS93] J.D. Moffett and M.S. Sloman, *Policy conflict analysis in distributed system management*, Journal of Organizational Computing (1993).

[MS04] Rajatish Mukherjee and Sandip Sen, *Towards a pareto-optimal solution in general-sum games*, 2004, citeseer.nj.nec.com/591017.html.

[Mum91] J. Mumpower, *The judgment policies of negotiators and the structure of negotiation problems*, Management Science **37** (1991), no. 10, 1304 – 1324.

[MW93] J.-J.Ch. Meyer and R. Wieringa (eds.), *Deontic logic in computer science*, Wiley, Chichester, UK, 1993.

[NE98] Donald Nute and Katrin Erk, *Defeasible logic graphs: I. theory*, Decision Support Systems **22** (1998), no. 3, 277–293.

[NHH98] Donald Nute, Zachary Hunter, and Christopher Henderson, *Defeasible logic graphs: II. implementation*, Decision Support Systems **22** (1998), no. 3, 295–306.

[NMB90] Donald Nute, Robert I. Mann, and Betty F. Brewer, *Controlling expert system recommendations with defeasible logic*, Decision Support Systems **6** (1990), no. 2, 153–164.

[NS92] M. Nowak and K. Sigmund, *Tit-for-tat in heterogeneous populations*, Nature **355** (1992), 250–2.

[NS93] ———, *A strategy of win-stay lose-shift that outperforms tit-for-tat in the prisoner's dilemma game*, Nature **364** (1993), 56–8.

[Nut88] Donald Nute, *Defeasible reasoning and decision support systems*, Decision Support Systems **4** (1988), no. 1, 97–110.

[Ogd38] C. K. Ogden, *Basic English : A general introduction with rules and grammar*, K. Paul, Trench, Trubner, London, UK, 1938.

[OL93] K. L. Ong and Ronald M. Lee, *An abductive approach to detecting inconsistencies in bureaucratic systems*, monograph, Euridis, Erasmus University, Rotterdam, The Netherlands, 1993.

[Oli96a] Jim R. Oliver, *A machine learning approach to automated negotiation and prospects for electronic commerce*, Journal of Management Information Systems **13** (1996), no. 3, 83 – 112.

[Oli96b] ———, *On artificial agents for negotiation in electronic commerce*, Ph.D. thesis, University of Pennsylvania, The Wharton School, Department of Operations and Information Management, Philadelphia, PA, April 1996.

[Oli97a] ———, *Artificial agents learn policies for multi-issue negotiation*, International Journal of Electronic Commerce **1** (1997), no. 4, 49–88.

[Oli97b] ———, *Artificial agents learn policies for multi-issue negotiation*, International Journal of Electronic Commerce **1** (1997), no. 4, 49 – 88.

[Ong92] Kay Liang Ong, *A formal model for maintaining consistency of evolving bureaucratic policies: A logical and abductive approach*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[OP02] A. Osterwalder and Y. Pigneur, *An e-business model ontology for modeling e-business*, Proceedings of the 15th Bled Electronic Commerce Conference (Kranj, Moderna Organizaja), 2002, pp. 1–12.

[Ouc79] W. G. Ouchi, *A conceptual framework for the design of organizational control mechanisms*, Management Science **25** (1979), 833–848.

[OY98] Wanda Orlikowski and JoAnne Yates, *Genre systems: Structuring interaction through communicative norms*, Working paper CCS WP #205, Sloan WP #4030, Massachusetts Institute of Technology, Sloan School of Management, Cambridge, MA, July 1998, http://ccs.mit.edu/papers/CCSWP205/index.html, accessed 22 August 2004.

[PAHL94] R. G. Palmer, W. B. Arthur, J. H. Holland, and Blake LeBaron, *Artificial economic life: a simple model of a stock market*, Physica D **75** (1994), 264–274.

[Par90] Terence Parsons, *Events in the semantics of English: A study in subatomic semantics*, Current Studies in Linguistics, The MIT Press, Cambridge, MA, 1990, ISBN: 0-262-66093-8.

[Par97] Frank Partnoy, *Fiasco the inside story of a wall street trader*, Penguin Books, New York, 1997.

[PBR95] Arnold Picot, Christine Bortenlänger, and Heine Röhrl, *The automation of capital markets*, Journal of Computer-Mediated Commerce **1** (1995), no. 3. Available online at http://www.ascusc.org/jcmc/-vol1/issue3/picot.html.

[PD94] N.W. Paton and O. Diaz, *Active database systems*, ACM Computing Surveys **1** (1994), no. 31.

[Pet62] C.A. Petri, *Kommunikation mit automaten*, Ph.D. thesis, University of Bonn, Bonn, Germany, 1962.

[Pet81] J.L. Peterson, *Petri net theory and the modeling of systems*, Prentice-Hall, 1981.

[PG03] A. G. Pateli and G. M. Giaglis, *A framework for understanding and analysing e-business models*, Proceedings of the 16th Bled Electronic Commerce Conference: eTransformation (Kranj, Moderna Organizaja) (R.T. Wigand, Y.H. Tan, J. Gricar, T. Pucihar, and T. Lunar, eds.), 2003, pp. 329–348.

[PKT01] O. Petrovic, C. Kittl, and R. D. Teksten, *Developing business models for e-business*, Proceedings of the 2nd International Conference on Electronic Commerce, 2001.

[Pol88] Livia Polanyi, *A formal model of the structure of discourse*, Journal of Pragmatics **12** (1988), 601–638.

[Pro03] Prolingua, *Operational communications, controlled languages, computing*, World Wide Web page, Accessed 28 February 2003, `http://www.prolingua.co.uk/`.

[PS97] H. Prakken and M. Sergot, *Defeasible deontic logic: Essays in nonmonotonic normative reasoning*, Synthese Library, vol. 263, ch. Dyadic Deontic Logic and Contrary-to-Duty Obligations, pp. 223–262, Kluwer Academic Publishers, 1997.

[PT02] Balaji Padmanabhan and Alexander Tuzhilin, *Knowledge refinement based on the discovery of unexpected patterns in data mining*, Decision Support Systems **33** (2002), no. 3, 309–321.

[PV00] F. Pianesi and A. Varzi, *Events and event talk: An introduction*, Speaking of Events (J. Higginbotham, F. Pianesi, and A. Varzi, eds.), Oxford University Press, Oxford UK, 2000, pp. 3–49.

[RC65] Anatol Rapoport and Albert M. Chammah, *Prisoner's dilemma: A study in conflict and cooperation*, The University of Michigan Press, Ann Arbor, MI, 1965.

[RE95] Alvin E. Roth and Ido Erev, *Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term*, Games and Economic Behavior **8** (1995), 164–212.

[Res75] N. Rescher, *A theory of possibility*, University of Pittsburgh Press, 1975.

[RGG76] Anatol Rapoport, Melvin J. Guyer, and David G. Gordon, *The 2×2 game*, The University of Michigan Press, Ann Arbor, MI, 1976.

[Ric43] I. A. Richards, *Basic English and its uses*, W. W. Norton & Company, Inc., New York, NY, 1943.

[RJB99] James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language reference manual*, Addison-Wesley, 1999.

[RL93] Young Ryu and Ronald M. Lee, *Formal representation of normative systems: A defeasible deontic reasoning approach*, monograph, Euridis, Erasmus University, Rotterdam, The Netherlands, 1993.

[RN95] S. Russell and P. Norvig, *Artificial intelligence*, Prentice Hall, New Jersey, 1995.

[RO77] R. R. Rosenberg and W. G. Ott, *College business law*, Shaum's Outline Series, McGraw-Hill, New York, NY, 1977.

[Roe92] Stephen F. Roehrig, *Probabilistic and defeasible reasoning using extended path analysis*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1992, Available as a technical report from the Department of Operations and Information Management.

[RS97] A. Rangaswamy and R. Shell, *Using computers to realize joint gains in negotiations: Toward and electronic bargaining table*, Management Science **43** (1997), no. 8, 1147.

[RSB79] Amnon Rapoport, William E. Stein, and Graham J. Burkheimer, *Response models for detection of change*, D. Reidel Publishing Company, Dordrecht, Holland, 1979.

[RTv96] J. Raskin, Yao-Hua Tan, and L. van der Torre, *How to model normative behavior in Petri Nets*, Proceedings of the 2nd Modelage Workshop on Formal Models of Agents (Sesimbra, Portugal), Modelage'96, 1996, pp. 223–241.

[RU71] N. Rescher and A. Urquhart, *Temporal logic*, Springer-Verlag, 1971.

[Rus05] Bertrand Russell, *On denoting*, Mind (1905), 479–93, Available at: http://www.mnstate.edu/gracyk/courses/web%20publishing/-russell_on_denoting.htm, accessed 30 July 2004.

[Rus96] John Rust, *Dealing with the complexity of economic calculations*, 1996, Workshop on Foundamental Limits to Knowledge in Economics.

[Ryu92] Young U. Ryu, *A formal representation of normative systems: A defeasible deontic reasoning approach*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1992.

[Ryu98] ———, *A logic-based modeling of resource consumption and production*, Decision Support Systems **22** (1998), no. 3, 243–257.

[Sag75] Naomi Sager, *Sublanguage grammers [sic] in information processing*, Journal of the American Society for Information Science **26** (1975), no. 1, 10–16.

[Sag86] ———, *Sublanguage: Linguistic phenomenon, computational tool*, Analyzing Language in Restricted Domains: Sublanguage Description and Processing (Ralph Grishman and Richard Kittredge, eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1986, pp. 1–17.

[Sal95] Airi Salminen, *EDIFACT for business computers: Has it succeeded?*, StandardView **3** (March 1995), no. 1, 33–42.

[San99] Tuomas Sandholm, *eMediator: A next generation electronic commerce server*, AAAI Workshop Technical Report WS-99-01 (Orlando, Fl), AAAI Workshop on AI in Electronic Commerce, 1999, pp. 46–55.

[SB98] Richar S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, The MIT Press, Cambridge, MA, 1998.

[SC95] T. Sandholm and R. Crites, *Multiagent reinforcement learning in iterated prisoner's dilemma*, Biosystems **37** (1995), 147–166, Special Issue on the Prisoner's Dilemma.

[SC96] Ira A. Smith and Philip R. Cohen, *Toward a semantics for an agent communication language based on speech acts*, Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vol. 2 (Menlo Park, California) (Howard Shrobe and Ted Senator, eds.), AAAI Press, 1996, pp. 24–31.

[Sch96] T. Schäl, *Workflow management systems for process organizations*, Springer Verlag, June 1996.

[Sco88] Peter C. Scott, *Requirements analysis assisted by logic modeling*, Decision Support Systems **4** (1988), no. 1, 17–25.

[Sea69] John R. Searle, *Speech acts*, Cambridge University Press, Cambridge, England, 1969.

[SEC99] Securities and Exchange Commission, *Securities and Exchange Commission special study: On-line brokerage: Keeping apace of cyberspace*, November 1999, Available online at http://www.sec.gov/news/studies/cyberspace.htm.

[SF90] Shimon Schocken and Tim Finin, *Meta-interpreters for rule-based inference under uncertainty*, Decision Support Systems **6** (1990), no. 2, 165–181.

[Sha89] C. Shapiro, *The theory of business strategy*, Rand Journal of Economics **20** (1989), 125–137.

[Sin93] Munidar P. Singh, *A semantics for speech acts*, Annals of Mathematics and Artificial Intelligence **8** (1993), no. I–II, 47–71, Reprinted in [HS98].

[Sin98] Munindar P. Singh, *Agent communication languages: Rethinking the principles*, IEEE Computer **31** (1998), no. 12, 40–47.

[Sky96] Brian Skyrms, *Evolution of the social contract*, Cambridge University Press, Cambridge, UK, 1996.

[Sky01] ———, *The stag hunt*, World Wide Web, 2001, Proceedings and Addresses of the American Philosophical Association. www.lps.uci.edu/home/fac-staff/faculty/skyrms/StagHunt.pdf. Accessed September 2004.

[SNY01] Olga Streltchenko, Nanjangud C. Narendra, and Yelena Yesha, *A reference architecture for multi-agent simulation of derivatives markets*, Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Applications (Bangor, Wales), 2001.

[Sow84] John F. Sowa, *Conceptual structures: Information processing in mind and machine*, Addison-Wesley, Reading, MA, 1984.

[Sow00] ———, *Knowledge representation: Logical, philosophical, and computational foundations*, Brooks/Cole, 2000.

[Ste89] John Sterman, *Modeling managerial behavior misperceptions of feedback in a dynamic decision making experiment*, Management Science **35** (1989), no. 3, 321–339.

[Ste94] K. Steel, *Another approach to standardising EDI*, Electronic Markets **12** (1994), 34–47.

[Ste96] Ken Steel, *The standardisation of flexible EDI messages*, Electronic Commerce: Current Research Issues and Challenges (Nabil R. Adam and Yelena Yesha, eds.), Springer-Verlag, Berlin, Germany, 1996, ISBN 3-540-60738-2., pp. 13–26.

[Ste04] John D. Sterman, *Teaching takes off: Flight simulators for management education*, http://web.mit.edu/jsterman/www/SDG-/beergame.html, 2004.

[Str59] P. F. Strawson, *Individuals*, Anchor Books, 1959.

[Str71] ———, *Logico-linguistic papers*, ch. On Referring, Methuen, 1971, Originally published 1957.

[Sut91] J. Sutton, *Sunk costs and market structure: Price competition, advertising, and the evolution of concentration*, The MIT Press, Cambridge, MA, 1991.

[SV85] John R. Searle and Daniel Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, Cambridge, England, 1985.

[SV99] R. Sarin and F. Vahid, *Payoff assessments without probabilities: A simple dynamic model of choice*, Games and Economic Behavior **28** (1999), 294–309.

[SV01] ———, *Predicting how people play games*, Games and Economic Behavior **34** (2001), 104–122.

[Swa80] Charles Swanland, *Basic English for science & technology*, Intercultural Press, Chicago, IL, 1980.

[TB99] C.P. Thorpe and J.C.L. Bailey, *Commercial contracts*, Kogan Page Limited, 1999.

[Tes02] Leigh Tesfatsion, *Notes on the Santa Fe Artificial Stock Market model*, Available online at http://www.econ.iastate.edu/classes/econ308x/-tesfatsion/sfistock.htm, 2002.

[Tie86] P.M. Tiersma, *The language of offer and acceptance: Speech acts and the question of intent*, California Law Review **74** (1986), 189–232.

[TK98] Gerald J. Tesauro and Jeffrey O. Kephar, *Foresight-based pricing algorithms in an economy of software agents*, Proceedings of International Conference on Information and Computation Economies, October 1998.

[TT99] Yao-Hua Tan and Walter Thoen, *A logical model of directed obligations and permissions to support electronic contracting*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 87–104.

[TT98a] ———, *A logical model of transfer obligations in trade contracts*, Accounting, Management and Information Technologies **8** (1998), no. 1, 23–38.

[TT98b] ———, *Towards a generic model of trust for electronic commerce*, International Journal of Electronic Commerce **3** (1998), no. 1, 65–81.

[TT01] ———, *A survey of electronic contracting related developments*, Proceedings of the 14th Bled Electronic Commerce Conference (Kranj, Moderna Organizaja), 2001.

[TT02] ———, *Formal aspects of a generic model of trust for electronic commerce*, Decision Support Systems **33** (2002), no. 3, 233–246.

[TTL00] D. Tapscott, D. Ticoll, and A. Lowy, *Digital capital: Harnessing the power of business webs*, Nicholas Brealy Publishing, London, UK, 2000.

[TWL00] M. Tu, E. Wolff, and W. Lamersdorf, *Genetic algorithms for automated negotiations: a FSM-based application approach*, Proceedings of 11th International Conference on Database and Expert Systems (DEXA 2000), 2000.

[Uhl90] Gerald R. Uhlich, *Descriptive theories of bargaining*, Springer-Verlag, Berlin, Germany, 1990.

[UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios, *The enterprise ontology*, The Knowledge Engineering Review **13** (1998), no. 1, 31–89.

[V+97] E. Garzòn Valdés et al. (eds.), *Normative systems in legal and moral theory – festschrift for Carlos E. Alchourrón and Eugenio Bulygin*, Duncker & Humblot, Berlin, Germany, 1997.

[Var98] Hal Varian, *Effect of the internet on financial markets*, 1998, Available online at http://www.sims.berkeley.edu/∼hal/Papers/brookings-paper.html.

[Var03] Hal R. Varian, *Intermediate microeconomics: A modern approach*, W. W. Norton & Company, New York, NY, 2003.

[vF71] B.C. van Fraasen, *Formal semantics and logic*, Macmillan, 1971.

[vR96] Victor Emil van Reijswoud, *The structure of business communication: Theory, model and application*, Ph.D. thesis, Technische Universiteit Delft, Delft, The Netherlands, June 1996, ISBN: 90-9009439-3. Address: Victor E. van Reijswoud, Juffrouw Idastraat 1, 2513 BE Den Haag, The Netherlands.

[VW65] G.H. Von Wright, *And next*, Acta Philosophica Fennica **Fasc. XVIII** (1965), 293–301.

[VW67] ———, *The logic of action – a sketch*, The Logic of Decision and Action (N. Rescher, ed.), University of Pittsburgh Press, 1967, pp. 121–136.

[VW68] ———, *An essay in deontic logic and the general theory of action*, Acta Philosophica Fennica **Fasc. XXI** (1968).

[vWd02] F. van der Poll, H. Weigand, and A. de Moor, *Communication diagnosis of a financial service process*, Proceedings of the Seventh International Workshop on the Language-Action Perspective on Communication Modelling (Delft, The Netherlands), LAP-2002, June 12–13, 2002, pp. 118–134.

[WD92] C.J.C.H. Watkins and P. Dayan, *Q-learning*, Machine Learning **8** (1992), 279–292.

[Wd03] Hans Weigand and Aldo de Moor, *Workflow analysis with communication norms*, Data & Knowledge Engineering **47** (2003), 349–369.

[WF87] Terry Winograd and Fernando Flores, *Understanding computers and cognition: A new foundation for design*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

[WGJS88] Fred Weeks, Alan Glover, Edward Johnson, and Peter Strevens, *Seaspeak training manual: Essential English for international maritime use*, Pergammon Press, Oxford, UK, 1988, Available via http://www.maritimeusa.com.

[Wid88] George R. Widmeyer, *Logic modeling with partially ordered preferences*, Decision Support Systems **4** (1988), no. 1, 87–95.

[Wid90] ———, *Reasoning with preferences and values*, Decision Support Systems **6** (1990), no. 2, 183–191.

[Win97] Hung Wing, *Managing complex, open, web-deployable trade objects*, Ph.D. thesis, University of Queensland, Department of Computer Science and Electrical Engineering, The University of Queensland, Brisbane Qld 4072, Australia, December 1997.

[Wit21] Ludwig Wittgenstein, *Tractatus logico-philosophicus*, Routledge and Keegan Paul, 1921, English translation by D.F. Pears & B.F. McGuinness, 1961.

[Wit58] ———, *Philosophical investigations*, third ed., Macmillan, New York, NY, 1953/1958, Translated by G.E.M. Anscombe.

[WJK00] Michael Wooldridge, Nicholas Jennings, and D. Kinny, *The gaia methodology for agent-oriented analysis and design*, Journal of Autonomous Agents and Multi-Agent Systems **3** (2000), no. 3, 285–312.

[Wri68] G. H. von Wright, *An essay in deontic logic and the general theory of action*, Acta Philosophica Fennica **Fasc. XXI** (1968).

[WS02] D. J. Wu and Yanjun Sun, *Cooperation in multi-agent bidding*, Decision Support Systems **33** (2002), no. 3, 335–347.

[WSdMD03] Hans Weigand, Mareike Schoop, Aldo de Moor, and Frank Dignum, *B2b negotiation support: The need for a communication perspective*, Group Decision and Negotiation **11** (2003), no. 1, 3–29.

[Wu97] D. J. Wu, *Using genetic algorithms to determine near-optimal pricing, investment and operating strategies in the electric power industry*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA, 1997.

[Wu00] ———, *Artificial agents for discovering business strategies for network industries*, International Journal of Electronic Commerce **5** (2000), no. 1, 9–36.

[Wv99] Hans Weigand and Willem-Jan van den Heuvel, *Meta-patterns for electronic commerce transactions based on the formal langauge for business communication (FLBC)*, International Journal of Electronic Commerce **3** (1998–99), no. 2, 45–66.

[Wv02] ———, *Cross-organizational workflow integration using contracts*, Decision Support Systems **33** (2002), no. 3, 247–265.

[XML98] The XML/EDI Group, *Home page for the XML/EDI Group*, Web page, January 1998, `http://www.xmledi.net`.

[Zar99] Frank G. Zarb, *The coming global digital stock market, speech at the National Press Club*, 1999, Available online at http://www.nasdaqnews.com/views/speech/digmarkets.htm.

[ZJW00] Franco Zambonelly, Nicholas Jennings, and Michael Wooldridge, *Organizational abstractions for the analysis and design of multi-agent systems*, Proceedings of the Firstst International Workshop on Agent-Oriented Software Engineering (Limerick, Ireland), 2000, pp. 127–141.

[ZKW02] Fang Zhong, Steven O. Kimbrough, and D. J. Wu, *Cooperative agent systems: Artificial agents play the ultimatum game*, Journal of Group Decision and Negotiation **11** (November 2002), no. 6, 433–447.

Index

Aalst, W.M.P., van der, 19
Abrahams, Alan S., 20, 23, 33–77
ACL, *see* agent communication languages
Adams, Fred, 20
agent communication languages, 22, 297–324, 334
– FIPA's ACL, 343
– FLBC, 297–324, 343
– KQML, 22, 334, 343
agent-based modelling, 248, 363–391
agents, 464
Andersen, K.A., 21
artificial agents, 463–475, 477–493, 509–527
– AAA, 445–461
– AAAA, 5
– adaptive, 445–461
– and markets, 393–419
– and supply chains, 363–391
auditing, 36

Bacon, Jean M., 23, 33–77
bargaining set, *see* games, bargaining set
battle of the forms, 185
Beethoven
– Ninth Symphony, 146
Belzer, Marvin, 20
Bhargava, Hemant K., 19, 21, 22
Bieber, Michael, 21
bill of lading, 191
Blanning, Robert W., 21
Bled Electronic Commerce Conference, 19
BNF, 209
Boisot, Max, 247–294
Bons, Roger W.H., 22, 198
Brewer, Betty F., 21
business process
– modelling, 36
– security and integrity, 36

Café Carlisle, 310
Cambridge Commonwealth Trust, 73
CANDID, 5
Casablanca, 317
Cathomen, Ivo, 21
Causey, Robert L., 21
CEFIC, 195
Chajadine, Sofia, 509–527
Chavez, Alex K., 421–443
Chen, Kuo-Tay, 19
Choudhary, Vidyanand, 22
coalition
– formation, 421–443
– value, 423
communicative action, 79
compositionality, 209
computational discovery, 7
conceptual graph, 89
contract, *see* provisions, 33
– enforcement, 61–64
– monitoring compliance, 231–246
– object, 147
– objects, 146
control mechanisms, 231–246
– definition, *232*
conventions
– signalling, 325–340
conversation, 343–360, 493
coordination, 33
counts-as, 49, 325–340
Covington, Michael A., 21, 22

Darwin, Charles, 4
Daskalopulu, Aspassia, 20, 23
de Moor, Aldo, 79–99
de Moore, Aldo, 23
Dean, James, 512
declarative, *see* representation, declarative
defeasible reasoning, 66, 197
Demaagd, Kurt, 363–391
DEMO, *see* Dynamic Essential Modeling of Organizations

DEON, International Workshop on Deontic Logic in Computer Science, 19
deontic concepts, *see* power, institutional, 37–39, 47–59, 131, 148, 231–246, 325–340
– conflicting obligations, 40
– duties, 34
– forbiddance, 33
– liability, 35
– norms, 35, 38, 82–86
– obligation, 33–35
– permission, 33–35
– prohibition, 35
– rights, 34
– soll, 86
deontic logic, 34, 39
description
– distributed definite, *310*
Dewitz, Sandra D., 19, 21, 198
Dignum, Frank, 22, 23
Dimitrakos, Theo, 23
disquotation theory, 34, *45–46*
distributed definite description, *310*, 312–314
documentary procedure, 185
documents
– contract, 35
– legal requirements, 35
– policy, 35
DTD, 201–225
duties, *see* deontic concepts, duties
Dynamic Essential Modeling of Organizations, 80, 81

e-commerce, 177, 445–461
ebXML, 194
Edee, 33–77
EDI, 201–225
EDI, electronic data interchange, 177, 187, 194
– ANSI X.12, 195
– UN/EDIFACT, 195
EDI, electronic document interchange, 178
EDIFACT, 202
EDIFICE, 195
electricity markets, 477–492
electronic commerce, 177
electronic contracting, 6
Electronic Data Interchange, 6
electronic data interchange, *see* EDI, 6, 321
equilibrium
– Cournot, 487
– Nash, 487, 501
Erk, Katrin, 22
Eun, Hyung, 247–294
EURIDIS, 18, 359
event semantics, 34, 37, 40, 211
– thematic roles, 43
evolution, 4
evolutionary computation, 363–391, 445–461
explanation
– procedural, 3
Eyers, David , 33–77

Fang, Christina, 20, 23
financial instruments, 167
– convertibles, 175
– debt, 171
– easements, 171
– equity, 173
– insurance, 170
– leases, 167
– licenses, 171
– options, 168
Finin, Timothy, 21, 393–419
finite state machines, 345
first trade problem, 202
FLBC, *see* agent communication languages, 325–340
forbid, *see* deontic concepts, forbiddance
forbiddance, *see* deontic concepts, forbiddance
formalisms
– graphs, 2
– logic, 2
– non-standard, 2
– procedures, 2
– standard, 2

game theory
– algorithmic, 478
– behaviioral, 478
– Folk Theorem, 466

games, 5, 363–391, 421–443, 463–475, 477–507
– agents, 493
– bargaining set, 423, 425
– characteristic function, 423
– Chicken, 509–527
– cooperative, 423
– Cournot, 477–492
– Exchange, 494
– Flag Chicken, 510, 520–524
– Iterated Prisoner's Dilemma, 483, 494, 495
– Prisoner's Dilemma, 480, 494, 509–527
– quota, 424
– replicator dynamics, 509–527
– Stag Hunt, 509–527
– Stochastic Flag Chicken, 516
– Stochastic Temporal Chicken, 516
Geerts, P., 21
generosity, 518
genetic algorithms, 445–461
genetic programming, 363–391, 449
Geyer, Georg, 21
Gordon, Michael, 391
granularity, 509–527
– defined, 509

Hammett, Dashiel, 313
Han, Kyeong Seok, 247–294
Henderson, Christopher, 22
Herrestad, Henning, 19
HICSS, Hawaii International Conference on System Sciences, 2, 18
Hooker, J.N., 20, 21
HTML, 203
Hua, Hua, 19, 22
Hunter, Zachary, 22
Hydra, 43

IBM Optimization Solutions and Library, 413
ICAIL, International Conference on Artificial Intelligence and Law, 19
ICC, International Chamber of Commerce, 185, 188
imperative, *see* representation, imperative
information objects, 147
information retrieval, 309
Isakowitz, Tomás, 21

Jennings, Nicholas R., 23
Jeroslow, Robert G., 20
Johnson, Edward, 302
Jones, Andrew J.I., 23, 325–342

KAoS, 345
Kartseva, Vera, 231–246
Kimbrough, Steven O., V–VI, 1–29, 73, 198, 201–227, 297–342, 391, 463–475, 477–492, 506
Klein, Stefan, 21
Kleinman, Gary, 24
knowledge management, 248
KQML, *see* agent communication languages, 334
Krishnan, Ramayya, 21, 22
Krogh, Cristen, 20
Kuhn, Christoph, 21
Kuiper, Ruurd, 22
Kuo, Ann, 463–475

λ abstraction, 108
Language/Action Perspective, 79, 301
languages
– artificial, 298
– designed, 298
– restricted, 298
– special, *298*
– sublanguages, 299
– telegraphic, 299
LAP, *see* Language/Action Perspective
LAP, Language-Action Perspective on Communication Modelling, International Working Conference, 19
learning
– aspiration level, 425
– associative, 463
– in games, 463–475, 477–507
– policy space, 463–475, 477–492
– Q-learning, 463–475, 477–507
– reinforcement, 421–443, 463–475, 477–507
– state space, 463–475, 477–492
– with incomplete information, 445–461

Lee, Picheng, 24
Lee, Ronald M., 18–23, 101–143, 145–198, 359
– CANDID, 5
legal
– entity, 150
legal processes, 183
legal reasoning, 33, 35
liability, *see* deontic concepts, liability
Lindemann, Markus A., 22
loans, 161
– of money, 162
Loewer, Barry, 20
logic
– action, 325–340
– counts-as, 325–340
– deontic, 131, 182, 231–246, 325–340
– free, 313
– modal, 131, 325–340
– special, 7
– *stit* (sees to it that), 338
logic programming, 2
Lomuscio, Alessio R., 20, 23
Lougee-Heimer, Robin, 18
Lu, Ming, 463–475, 477–492

Mack, Daniel, 509–527
MacMillan, Ian, 247–294
Maibaum, Tom, 23
Maltese Falcon, The, 313
Mann, Robert I., 21
markets
– auction, 395
– dealer, 395
– hybrid, 395
memory
– working, 499
message marker, 311
messenger model, 195
microFLBC, 203
Microsoft, 204
Microsoft Research Cambridge, 73
money, 146
Montague semantics, 5
Moore, Scott A., 18, 19, 22, 23, 343–360, 363–391
Müller, Rudolf, 21
Murphy, Frederic, 477–492

NASDAQ, 396
Nash equilibrium, 501
negotiation
– automated, 445–461
New York Stock Exchange, 395
norms, *see* deontic concepts, norms
– communicative, 80
Nute, Donald, 20–22

obligation, *see* deontic concepts, obligation
oblige, *see* deontic concepts, obligation
occurrence, 34, 35, 38, *40–45*, 47–59
Oliver, Jim R., 19, 21, 445–461
organizations, 79–98
ownership, 148

Pace, Stefano, 23
Padmanabhan, Balaji, 23
Palmon, Dan, 24
Pareto frontier, 501
peer-to-peer networking, 231
Peirce, Charles Sanders, 301
performative, 179
– document, 146
performative communications, 177
performative network, 186
permission, *see* deontic concepts, permission
permit, *see* deontic concepts, permission
Petri Nets, 71, 177, 192, 236
– Documentary, 192
Philadelphia, 310
pizza, 88
policy, *see* provisions
– specification, 39
possession, 148
power
– institutional, 34, 35, 37, 49, 55
price discovery mechanism, 395
Prince, 310
– the artist formerly known as, 310
problems
– dynamic reference fixing, 301, 307–312
– speech act representation, 301
procedures, 34
programming languages, 33

prohibit, *see* deontic concepts, prohibition
prohibition, *see* deontic concepts, prohibition
Prolog, 338
promissory objects, 154
propositional content, 33
provisions, *see* deontic concepts, 37
purchase order, 215–218

questions
– w (who, what, where, when,why), 319
– yes-no, 320

reasoning
– choice, 429
– counterfactual, 427
Rebel without a Cause, 512
reciprocity, 238
regulation, *see* provisions
reinforcement learning, *see* learning, reinforcement
representation
– declarative, 33
– imperative, 34
– of contracts, 35
– of legal requirements, 35
– of policies, 35
– procedural, 37
requirements
– user, 36
rights, *see* deontic concepts, rights
Roehrig, Stephen F., 19
Rons, Roger W.H., 20
rules, 34
Ryu, Young U., 21, 22

Santa Fe Institute, 437
Schmid, Beat F., 21, 22
Schocken, Shimon, 21
Schoop, Mareike, 23
Scott, Peter C., 20
SeaSpeak, 297–324
– and Edward Johnson, 302
– message marker, *302*
Securities and Exchange Commission, 151
security, 189
semantic network, 89
semantic transparency, 202
semantics
– Montague, 5
Sergot, Marek, 18, 23
SGML, 204
Shakun, Melvin F., VI, 18
Shand, Brian, 73
Shaw, Mike, VI
Short, Bobby, 310
Sim-I-Space, 247–294
Slan, Aaron Jeffrey, 509–527
small and medium-sized enterprises, *see* SMEs
smart card technology, 191
SMEs, 17, 202
social action, 177
social dilemmas, 490
social networks, 250
spanning problem, 201
speech acts, 6, 34, 177, 182, 211, 325–340
– advice, 303
– and intentions, 334
– $F(P)$ framework, 301
– illocutionary force, 301
– information, 303–307
– instruction, 303
– intention, 303
– propositional content, 301
– question, 303
– request, 303
– warning, 303
spontaneity problem, 202
Sprague, Ralph H., Jr., VI, 18
SPSS, 146
standardization, 396
strategic behavior, *see* games
strategic contexts, 5
strategies
– Tit-for-Tat, 494, 495, 504
– Tit-for-Two-Tats, 505
strategy formation, 7, 363–391
Streltchenko, Olga, 393–419
stylistic variant, *305*
Sun, Yanjun, 23
SuperDOT, 396
supply chains

– management, 363–391
Swarm, 248
SWIFT, 195
system and process modelling, 7

Tan, Casey, 247–294
Tan, Yao-Hua, 22, 23, 198, 231–246, 359
themes
– inference, 5
– learning, 5
– representation, 5
Thoen, Walter, 22, 23
time-sensitive transactions, 60
topics
– computational discovery, 7
– electronic contracting, 6
– electronic data interchange, 6
– first trade problem, 6
– special logics, 7
– speech acts, 6
– strategy formation, 7
– system and process modelling, 7
trade facilitation, 195
trust, 231–246, 493–507
– party versus control, 232
trusted third party, 191
Tuzhilin, Alexander, 23

UK Overseas Research Students Scheme, 73
UML, 79, 344
UML, Unified Modeling Language, 3
UN/EDIFACT, 6
UNCTAD, 195
University of Cape Town Postgraduate Scholarships Office, 73
University Scholars Fund at the University of Pennsylvania, 437

Valluri, Annapurna, 23
van den Heuvel, Willem-Jan, 22, 23
van Reijswoud, Victor Emil, 19
VAN, Value Added Network, 190
Vermeir, D., 21

Wagenaar, René, 198
Wallace, Alfred Russell, 4
waybill, 191
web services, 231
Weigand, Hans, 22, 23, 79–99
Weinberg, Paul, 18
Wharton-SMU Research Center of Singapore Management University, 263
Whinston, Andrew, VI, 18
Widmeyer, George R., 20, 21
Wing, Hung, 20
Wooldridge, Michael, 23
WordNet, 219
workflow modelling, 79–98
World Wide Web, 204
Wrigley, Clive, 198
Wu, D.J., V–VI, 1–29, 391, 506
– Dongjun, 20

X12, 6, 202
XML, 201–225

Yang, Yinghui (Catherine), 297–324
Yesha, Yelena, 393–419

Zheng, Zhiqiang, 23
Zhong, Fang, 23, 391, 493–507
Zwass, Vladimir, VI, 18