Alexander Artikis
Gregory M.P. O'Hare
Kostas Stathis
George Vouros (Eds.)

Engineering Societies in the Agents World VIII

LNAI 4995

8th International Workshop, ESAW 2007
Athens, Greece, October 2007
Revised Selected Papers

Springer

Lecture Notes in Artificial Intelligence 4995

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Alexander Artikis Gregory M.P. O'Hare
Kostas Stathis George Vouros (Eds.)

Engineering Societies in the Agents World VIII

8th International Workshop, ESAW 2007
Athens, Greece, October 22-24, 2007
Revised Selected Papers

Springer

Volume Editors

Alexander Artikis
National Centre for Scientific Research "Demokritos"
Software & Knowledge Engineering Laboratory
Athens 15310, Greece
E-mail: a.artikis@acm.org

Gregory M.P. O'Hare
University College Dublin, Adaptive Information Cluster
Belfield, Dublin 4, Ireland
E-mail: gregory.ohare@ucd.ie

Kostas Stathis
Royal Holloway, University of London, Dept. of Computer Science
Egham, Surrey, TW20 0EX, UK
E-mail: kostas@cs.rhul.ac.uk

George Vouros
University of the Aegean
Dept. of Information and Communication Systems Engineering
83200 Samos, Greece
E-mail: georgev@aegean.gr

Library of Congress Control Number: 2008936469

CR Subject Classification (1998): I.2.11, D.2, K.4, D.1.3, H.3.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-87653-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-87653-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12465071 06/3180 5 4 3 2 1 0

Preface

The 8th annual international workshop "Engineering Societies in the Agents' World" was hosted by the National Centre for Scientific Research "Demokritos", in Athens, Greece, in October 2007. The workshop was organized as a stand-alone event, running over three days. ESAW 2007 built upon the success of prior ESAW workshops: ESAW 2006 held in Dublin, ESAW 2005 held in Kuşadasi, going back to the first ESAW workshop, which was held in Berlin in 2000. ESAW 2007 was attended by 40 participants from 10 different countries. Each presentation was followed by highly interactive discussions, in line with the ESAW spirit of having open discussions with fellow experts.

The ESAW workshop series started in 2000 to provide a forum for presenting highly inter-disciplinary work on technologies, methodologies, platforms and tools for the engineering of complex artificial agent societies. Such systems have found applications in many diverse domains such as space flight operations, e-business and ambient intelligence. Despite ESAW traditionally placing emphasis on practical engineering issues and applications, the workshop did not exclude theoretical and philosophical contributions, on the proviso that they clearly documented their connection to the core applied issues.

Discussions coalesced around the following themes:

- electronic institutions;
- models of complex distributed systems with agents and societies;
- interaction in agent societies;
- engineering social intelligence in multi-agent systems;
- trust and reputation in agent societies;
- analysis, design and development of agent societies.

Three invited presentations underlined the interdisciplinary nature of research on agent societies by considering aspects of action and agency, and conflict detection and resolution in norm-governed multi-agent systems, and coalition formation for collective economic action in electronic markets. The first invited talk was given by Marek Sergot, a professor of computational logic at Imperial College London. In his talk, the contents of which appear in this volume as an invited submission, Professor Sergot presented a formal language for describing and analyzing norm-governed systems. The language provides constructs for expressing properties of states and transitions in a transition system. Moreover it includes modalities of the kind found in logics of action/agency for expressing the fact that an agent brings it about that, or is responsible for, its being the case that a certain property holds.

The second invited talk was given by Robert Axtell, a professor at the George Mason University, and an external professor at the Santa Fe Institute. Professor Axtell presented the conditions under which it is individually rational for agents to spontaneously form coalitions in order to engage in collective economic

action in e-commerce. He showed that, under certain conditions, self-organized coalitions of agents are capable of extracting welfare improvements even in non-cooperative environments.

The third invited talk was given by Tim Norman, a senior lecturer in the Department of Computing Science of the University of Aberdeen. Dr. Norman elaborated on three challenges concerning the development of Virtual Organizations. First, social norms governing the behavior of agents must be explicitly declared rather than being implicit in the design of a system. Second, it should be recognized that there are situations in which there is no possible course of action available for an agent that satisfies all norms. Third, agents must have mechanisms to resolve conflicts and to reason about norm violation.

The original contributions, the slides of the presentations, and more information about the workshop are available online at the ESAW 2007 website (`http://esaw07.iit.demokritos.gr`). The present post-proceedings continue the series published by Springer (ESAW 2000: LNAI 1972, ESAW 2001: LNAI 2203, ESAW 2002: LNAI 2577, ESAW 2003: LNAI 3071, ESAW 2004: LNAI 3451, ESAW 2005: LNAI 3963, ESAW 2006: LNAI 4457). This volume contains extended and substantially revised versions of selected papers from ESAW 2007 and an invited contribution by Marek Sergot.

The organization of ESAW 2007 would not have been possible without the financial help of:

- the University of the Aegean, Greece,
- the EU-funded project Argugrid,
- cosmoONE Hellas MarketSite,
- NCSR "Demokritos", Greece,
- the Hellenic Artificial Intelligence Society,
- the Institute for Human and Machine Cognition (IHMC), US, and
- Imperial College London, UK.

We would like to thank the Steering Committee for their guidance, the Program Committee and the additional reviewers for the insightful reviews, and the Local Organizing Committee for arranging an enjoyable event. We would also like to thank all the researchers who submitted a paper to the workshop. Finally, we would like to offer our thanks to Alfred Hofmann and the Springer crew for helping us realize these post-proceedings.

The next ESAW workshop will be hosted in France by the Ecole Nationale Supérieure des Mines de Saint-Etienne, in September 2008, with Alexander Artikis, Gauthier Picard and Laurent Vercouter as organizers. We look forward to even more lively interactions, and a still higher level of originality and innovation.

June 2008

Alexander Artikis
Gregory M. P. O'Hare
Kostas Stathis
George Vouros

Organization

ESAW 2007 Organizers

Alexander Artikis	NCSR "Demokritos", Greece
Gregory M.P. O'Hare	University College Dublin, Ireland
Kostas Stathis	Royal Holloway, University of London, UK
George Vouros	University of the Aegean, Greece

ESAW Steering Committee

Marie-Pierre Gleizes	IRIT Université Paul Sabatier, France
Andrea Omicini	Università di Bologna, Italy
Paolo Petta	Austrian Research Institute for Artificial Intelligence, Austria
Jeremy Pitt	Imperial College London, UK
Robert Tolksdorf	Free University of Berlin, Germany
Franco Zambonelli	Università di Modena e Reggio Emilia, Italy

ESAW 2007 Local Organizing Committee

Alexander Artikis	NCSR "Demokritos", Greece
George Giannakopoulos	NCSR "Demokritos", Greece
Dimosthenis Kaponis	Imperial College London, UK
Eugenia Pantouvaki	NCSR "Demokritos", Greece
Vassilis Spiliopoulos	NCSR "Demokritos", Greece
Ilias Zavitsanos	NCSR "Demokritos", Greece

ESAW 2007 Program Committee

Grigoris Antoniou	University of Crete, Greece
Federico Bergenti	Università di Parma, Italy
Carole Bernon	IRIT Université Paul Sabatier, France
Guido Boella	Università degli Studi di Torino, Italy
Olivier Boissier	Ecole Nationale Supérieure des Mines de Saint-Etienne, France
Jeff Bradshaw	IHMC, USA
Monique Calisti	Whitestein Technologies, Switzerland
Jacques Calmet	University of Karlsruhe, Germany
Cristiano Castelfranchi	ISTC-CNR, Italy
Luca Cernuzzi	Universidad Católica "Nuestra Señora de la Asunción", Paraguay

Helder Coelho	University of Lisbon, Portugal
Rem Collier	University College Dublin, Ireland
Dan Corkill	University of Massachusetts at Amherst, USA
R. Scott Cost	University of Maryland Baltimore County, USA
Aspassia Daskalopulu	University of Thessaly, Greece
Mehdi Dastani	Utrecht University, The Netherlands
Paul Davidsson	Blekinge Institute of Technology, Sweden
Keith Decker	University of Delaware, USA
Oguz Dikenelli	Ege University, Turkey
Riza Cenk Erdur	Ege University, Turkey
Rino Falcone	ISTC-CNR, Italy
Paul Feltovich	IHMC, USA
Jean-Pierre George	IRIT Université Paul Sabatier, France
Paolo Giorgini	University of Trento, Italy
Michael O'Grady	University College Dublin, Ireland
Frank Guerin	University of Aberdeen, UK
Salima Hassas	Université Claude Bernard Lyon 1, France
Lloyd Kamara	Imperial College London, UK
Anthony Karageorgos	University of Thessaly, Greece
Manolis Koubarakis	University of Athens, UK
Michael Luck	University of Southampton, UK
Fabien Michel	Université de Reims, France
Tim Miller	University of Liverpool, UK
Pavlos Moraitis	Paris-Descartes University, France
Pablo Noriega	IIIA, Spain
Sascha Ossowski	Universidad Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Juan Pavon Mestras	Universidad Complutense de Madrid, Spain
Paolo Petta	Austrian Research Institute for Artificial Intelligence, Austria
Jeremy Pitt	Imperial College London, UK
Alessandro Ricci	Università di Bologna, Italy
Giovanni Rimassa	Whitestein Technologies, Switzerland
Juan Antonio Rodriguez Aguilar	IIIA, Spain
Fariba Sadri	Imperial College London, UK
Maarten Sierhuis	RIACS/NASA Ames Research Center, USA
Tiberiu Stratulat	LIRMM, France
Robert Tolksdorf	Free University of Berlin, Germany
Leon Van der Torre	University of Luxembourg, Luxembourg
Luca Tummolini	ISTC-CNR, Italy
Paul Valckenaers	Katholieke Universiteit Leuven, Belgium
Wamberto Vasconcelos	University of Aberdeen, UK
Mirko Viroli	Università di Bologna, Italy

Marina De Vos	University of Bath, UK
Danny Weyns	Katholieke Universiteit Leuven, Belgium
Pinar Yolum	Bogazici University, Turkey
Franco Zambonelli	Università di Modena e Reggio Emilia, Italy

Additional Reviewers

Marc Esteva	IIIA, Spain
Ramón Hermoso	Universidad Rey Juan Carlos, Spain
Dimosthenis Kaponis	Imperial College London, UK
Jarred McGinnis	Royal Holloway, University of London, UK

Table of Contents

Electronic Institutions

Models of Complex Distributed Systems with Agents and Societies

Interaction in Agent Societies

Engineering Social Intelligence in Multi-agent Systems

Trust and Reputation in Agent Societies

Analysis, Design and Development of Agent Societies

Action and Agency in Norm-Governed Multi-agent Systems

Marek Sergot

Department of Computing, Imperial College London
London SW7 2AZ, UK
mjs@doc.ic.ac.uk

Abstract. There is growing interest in the idea that, in some cases, interactions among multiple, independently acting agents in a multi-agent system can be regulated and managed by norms (or 'social laws') which, if respected, allow the agents to co-exist in a shared environment. We present a formal (modal-logical) language for describing and analysing such systems. We distinguish between system norms, which express a system designer's view of what system behaviours are deemed to be legal, permitted, desirable, and so on, and agent-specific norms which constrain and guide an individual agent's behaviours and which are supposed to be incorporated, in one way or another, in the agent's implementation. The language provides constructs for expressing properties of states and transitions in a transition system, and modalities of the kind found in logics of action/agency for expressing that an agent brings it about that, or is responsible for, its being the case that A. The novel feature is that an agent, or group of agents, brings it about that a transition has a certain property rather than bringing it about that a certain state of affairs obtains, as is usually the case. The aim of the paper is to motivate the technical development and illustrate the use of the formal language by means of a simple example in which there there are both physical and normative constraints on agents' behaviours. We discuss some relationships between system norms and agent-specific norms, and identify several different categories of non-compliant behaviour that can be expressed and analysed using the formal language. The final part of the paper presents some transcripts of output from a model-checker for the language.

1 Introduction

There has been growing interest in recent years in norm-governed multi-agent systems. References to normative concepts (obligation, permission, commitment, social commitment, ...) feature prominently in the literature. One reason for this interest is clear, for there are important classes of applications, in e-commerce, contracting, trading, e-government, and so on, where the domain of application is defined by and regulated by laws, regulations, codes of practice, and standards of various kinds whose existence is an essential ingredient of any application. Another, somewhat different, motivation is the idea that, in some cases, agent interactions generally can best be regulated and managed by the use of norms. The

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 1–54, 2008.

term ‘social laws’ has also been used in this connection, usually with reference to ‘artificial social systems’. A ‘social law’ has been described as a set of obligations and prohibitions on agents’ actions, that, if respected, allow multiple, independently acting agents to co-exist in a shared environment. The question of what happens to system behaviour when norms or social laws are not respected, however, has received little or no serious attention. It is also not entirely clear from works in this area whether these norms are intended to express only the system designer’s view of what behaviours are legal, permitted, desirable, and so on, or whether they are supposed to be taken into account, explicitly or implicitly, in the implementation of the agents themselves, or both.

In a recent paper [1] we presented a formal framework, called there a ‘coloured agent-stranded transition system’, which adds two components to a labelled transition system. The first component partitions states and transitions according to various ‘colourings’, used to represent norms (or ‘social laws’), of two different kinds. *System norms* express a system designer’s point of view of what system states and system transitions are legal, permitted, desirable, and so on. A separate set of individual *agent-specific* norms are intended to guide or constrain an individual agent’s behaviours. They are assumed to be taken into account in the agent’s implementation, or in the case of deliberative agents with reasoning and planning capabilities, in the processes an agent uses to determine its choice of actions to be performed. The second component of a ‘coloured agent-stranded transition system’ is a way of picking out, from a global system transition representing many concurrent actions by multiple agents and possibly the environment, an individual agent’s actions, or ‘strand’, in that transition. This is to enable us to say that in a particular transition it is specifically one agent’s actions that are in compliance or non-compliance with a system or agent-specific norm rather than some other’s. This framework allowed us in turn to identify and characterise several different categories of non-compliant behaviour, distinguishing between various forms of unavoidable or inadvertent non-compliance, behaviour where an agent does ‘the best that it can’ to comply with its individual norms but nevertheless fails to do so because of actions of other agents, and behaviour where an agent could have complied with its individual norms but did not. The aim, amongst other things, is to be able to investigate what kind of system properties emerge if we assume, for instance, that all agents of a certain class will do the best that they can to comply with their individual norms, or never act in such a way that they make non-compliance unavoidable for others. The other general aim, which is to consider how agent-specific norms can be incorporated into an agent’s implementation, was not discussed. It is a topic of current work.

This paper presents a further development and refinement of those ideas. Specifically, we now prefer to separate the ‘colourings’ used to represent norms from the more general structure of an agent-stranded transition system. We present a formal (modal-logical) language for talking about properties of states and transitions, including but not restricted to their ‘colourings’, and for talking about agent strands of transitions. The language has operators for expressing

that a particular agent, or group of agents, *brings it about* that such-and-such is the case, in the sense that it is responsible for, or its actions are the cause of, such-and-such being the case. The resulting logic bears a strong resemblance to Ingmar Pörn's (1977) logic of 'brings it about' action/agency [2], except that we switch from talking about an agent's bringing about a certain state of affairs to an agent's bringing it about that a transition has a certain property. The general aim of the paper is to motivate the technical development and illustrate something of the expressiveness of the formal language. We use the same, rather simple, example discussed in the earlier paper [1] but present it now in terms of the new formal system. Technical details of the logic, comparisons with other works in the logic of action/agency, and discussion of various forms of collective or group agency are beyond the scope of this paper. These topics are covered elsewhere [3].

It is important to stress that we make *no assumptions* about the reasoning or perceptual capabilities of the agents. Agents could be deliberative (human or computer) agents, purely reactive agents, or simple computational devices. We make no distinction between them here. This is for both methodological and practical reasons. From the methodological point of view, it is clear that genuine collective or joint action involves a very wide range of issues, including joint intention, communication between agents, awareness of another agent's capabilities and intentions, and many others. We want to factor out all such considerations, and investigate only what can be said about individual or collective agency when all such considerations are ignored. The result might be termed 'a logic of unwitting (collective) agency'—'unwitting' means both inadvertent and unaware. The logic of unwitting agency might be extended and strengthened in due course by bringing in other considerations such as (joint) intention; we do not discuss any such possibilities here. From the practical point of view, there is clearly a wide class of applications for multi-agent systems composed of agents with reasoning and deliberative capabilities. There is an even wider class of applications if we consider also simple 'lightweight' agents with no reasoning capabilities, or systems composed of simple computational units in interaction. We want to be able to consider this wider class of applications too.

The formal language presented here has been implemented, in the form of a model-checker that can be used to evaluate formulas on a given transition system. It is included as part of the ICCALC system[1], which at its core is a re-implementation of the 'Causal Calculator' CCALC[2] developed at the University of Texas and made available as a means of performing computational tasks using the action language $\mathcal{C}+$. $\mathcal{C}+$ [4] is a formalism for defining transition systems of a certain kind. It provides a treatment of default persistence ('inertia'), non-deterministic and concurrent actions, and indirect effects of actions ('ramifications'). CCALC can be used (among other things) to generate (a symbolic representation of) a transition system defined by means of $\mathcal{C}+$ laws. ICCALC retains the core functionality of CCALC, and the core implementation techniques,

[1] http://www.doc.ic.ac.uk/~rac101/iccalc/
[2] http://www.cs.utexas.edu/users/tag/cc

and adds a number of other features, such as the ability to pass the transition system to standard CTL model checking systems (specifically NuSMV). ICCALC also supports a number of extended forms of $\mathcal{C}+$, of which the language $n\mathcal{C}+$ is the most relevant here. $n\mathcal{C}+$ [5,6] is an extended form of $\mathcal{C}+$ designed specifically for representing simple normative and institutional concepts. An action description in $n\mathcal{C}+$ defines a coloured (agent-stranded) transition system of a certain kind. The examples discussed in this paper are constructed by formulating them as $n\mathcal{C}+$ action descriptions, using ICCALC to generate (a symbolic representation of) the transition system so defined, and then passing the transition system to the model checker that evaluates formulas of the language presented in this paper. However, the framework presented in this paper is more general, and is *not* restricted to transition systems of the kind defined by $\mathcal{C}+$ or $n\mathcal{C}+$.

2 Labelled Transition Systems

2.1 Preliminaries

Transition systems. A labelled transition system (LTS) is usually defined as a structure $\langle S, A, R\rangle$ where

- S is a (non-empty) set of *states*;
- A is a set of *transition labels*, also called *events*;
- R is a (non-empty) set of labelled *transitions*, $R \subseteq S \times A \times S$.

When (s, ε, s') is a transition in R, s is the initial state and s' is the resulting state, or end state, of the transition. ε is *executable* in a state s when there is a transition (s, ε, s') in R, and *non-deterministic* in s when there are transitions (s, ε, s') and (s, ε, s'') in R with $s' \neq s''$. A *path* or *run* of length m of the labelled transition system $\langle S, A, R\rangle$ is a sequence $s_0\, \varepsilon_0\, s_1 \cdots s_{m-1}\, \varepsilon_{m-1}\, s_m \quad (m \geq 0)$ such that $(s_{i-1}, \varepsilon_{i-1}, s_i) \in R$ for $i \in 1..m$. Some authors prefer to deal with structures $\langle S, \{R_a\}_{a\in A}\rangle$ where each R_a is a binary relation on S.

It is helpful in what follows to take a slightly more general and abstract view of transition systems. A transition system is a structure $\langle S, R, prev, post\rangle$ where

- S and R are disjoint, non-empty sets of *states* and *transitions* respectively;
- *prev* and *post* are functions from R to S: $prev(\tau)$ denotes the initial state of a transition τ, and $post(\tau)$ its resulting state.

In this more abstract account, a *path* or *run* of length m of the transition system $\langle S, R, prev, post\rangle$ is a sequence $\tau_1 \cdots \tau_{m-1}\, \tau_m \quad (m \geq 0)$ such that $\tau_i \in R$ for every $i \in 1..m$, and $post(\tau_i) = prev(\tau_{i+1})$ for every $i \in 1..m{-}1$.

A *labelled transition system* (LTS) is a structure

$$\langle S, A, R, prev, post, label\rangle$$

where S, R, *prev*, and *post* are as above, and where *label* is a function from R to A. The special case of a LTS in which $R \subseteq S \times A \times S$ then corresponds to the

case where $prev(\tau) = prev(\tau')$ and $post(\tau) = post(\tau')$ and $label(\tau) = label(\tau')$ implies $\tau = \tau'$, and in which $prev((s, \varepsilon, s')) = s$, $post((s, \varepsilon, s')) = s'$, and $label((s, \varepsilon, s')) = \varepsilon$. The more abstract account is of little practical significance but is helpful in that it allows a more concise statement of some things we want to say about transition systems. It is also more general: transitions are not identified by (s, ε, s') triples—there could be several transitions with the same initial and resulting states and the same label. Nothing in what follows turns on this. Henceforth, we will write $\langle S, A, R\rangle$ as shorthand for $\langle S, A, R, prev, post, label\rangle$ leaving the functions *prev*, *post*, and *label* implicit.

Interpreted transition systems. Given a labelled transition system, it is usual to define a language of propositional 'fluents' or 'state variables' in order to express properties of states. Given an LTS $\langle S, A, R\rangle$ and a suitably chosen set of atomic propositions, a model is a structure $\mathcal{M} = \langle S, A, R, h^{\mathrm{f}}\rangle$ where h^{f} is a valuation function which specifies, for every atomic proposition p, the set of states in the LTS at which p is true.

We employ a *two-sorted* language. We have a set σ^{f} of propositional atoms for expressing properties of states, and a disjoint set σ^{a} of propositional atoms for expressing properties of events and transitions. Models are structures $\mathcal{M} = \langle S, A, R, h^{\mathrm{f}}, h^{\mathrm{a}}\rangle$ where h^{f} is a valuation function for atomic propositions σ^{f} in states S and h^{a} is a valuation function for atomic propositions σ^{a} in transitions R. We then extend this two-sorted propositional language with (modal) operators for converting state formulas to transition formulas, and transition formulas to state formulas. Concretely, where φ is a transition formula, the state formula $[\varphi]F$ expresses that the state formula F is satisfied in every state following a transition of type φ. The transition formulas $0{:}F$ and $1{:}G$ are satisfied by a transition τ when the initial state of τ satisfies state formula F and the resulting state of τ satisfies state formula G, respectively. The details are summarised presently.

It is not clear whether evaluating formulas on transitions in this fashion is novel or not. Große and Khalil [7] evaluate formulas on state-event pairs (s, ε) when the transition system is a set of triples (s, ε, s') but that is not the same as we have here. Venema [8] uses a two-sorted language for expressing properties of points and lines in projective geometry, though naturally the choice of modal operators is different there.

We also find it convenient to add a little more structure to the underlying propositional language. This is not essential but makes the formulation of typical examples clearer and more concise. It is also the propositional language that is supported by $\mathcal{C}+$ and $n\mathcal{C}+$, and the CCalc and iCCalc implementations.

Multi-valued signatures. The following is adapted from [4]. A *multi-valued propositional signature* σ is a set of symbols called *constants*. For each constant c in σ there is a non-empty set $dom(c)$ of values called the *domain* of c. For simplicity, in this paper we will assume that each $dom(c)$ is finite and has at least two elements. An *atom* of a signature σ is an expression of the form $c{=}v$ where c is a constant in σ and $v \in dom(c)$. A *formula* of signature σ is any truth-functional compound of atoms of σ.

A *Boolean* constant is one whose domain is the set of truth values $\{\mathsf{t}, \mathsf{f}\}$. If c is a Boolean constant, c is shorthand for the atom $c{=}\mathsf{t}$ and $\neg c$ for the atom $c{=}\mathsf{f}$. More generally, if c is a constant whose domain is $\{v_1, \ldots, v_n, \mathsf{f}\}$, then by convention we write $\neg c$ as shorthand for the atom $c{=}\mathsf{f}$.

An *interpretation* of a multi-valued signature σ is a function that maps every constant c in σ to some value v in $dom(c)$; an interpretation I *satisfies* an atom $c{=}v$ if $I(c) = v$. We write $\mathrm{I}(\sigma)$ for the set of interpretations of σ.

As observed in [4], a multi-valued signature of this type can always be translated to an equivalent Boolean signature. Use of a multi-valued signature makes the formulation of examples more concise.

Syntax and semantics

The base propositional language is constructed from a set σ^{f} of *state constants* (also known as 'fluents' or 'state variables') and a disjoint set σ^{a} of *event constants*. In previous work we followed the terminology of [4] and called the constants of σ^{a} 'action constants'. This terminology is misleading however. Although event constants *are* used to name actions and attributes of actions, they are also used to express properties of an event or transition as a whole. An example of an event constant might be $x{:}move$ with domain $\{l, r, \mathsf{f}\}$: the atom $x{:}move{=}l$ represents that agent x moves in direction l, $x{:}move{=}r$ that x moves in direction r, and $\neg x{:}move$ (which, recall, is shorthand for $x{:}move{=}\mathsf{f}$) that x does not move in a given transition. In ICCALC we employ an (informal) convention that event constants with a prefix 'x:' are intended to represent actions by an agent x. The (Boolean) event constant $falls(vase)$ might be used to represent transitions in which the object *vase* falls from a table to the ground (say). Here there is no prefix '*vase*:'—'falls' is not an action that is meaningfully performed by the object *vase*. Event constants are also used to express properties of a transition as whole, for instance, whether it is desirable or undesirable, timely or untimely, permitted or not permitted, and so on. For this reason we prefer the term 'event constant' for the elements of σ^{a}, and we reserve the term 'action constant' for referring informally to those event constants that are intended to represent actions by an agent. In general, an event (or transition label) will represent multiple concurrent actions by agents and the environment, concurrent actions, such as the falling of an object, that cannot be ascribed to any agent, and other properties of the event, such as whether it is desirable or undesirable, desirable or undesirable from the point of view of an agent x, timely or untimely, and so on.

For example, the formula

$$a{:}move{=}l \wedge \neg\, b{:}move{=}l \wedge \neg c{:}move \wedge falls(vase) \wedge trans{=}red$$

might represent an event in which a moves to the left, b does not move to the left, c does not move at all, and the object *vase* falls. The atom $trans{=}red$ might represent that the event is illegal (say), or undesirable, or not permitted.

Propositional formulas of σ^{a} are evaluated on transition labels/events. When an event satisfies a propositional formula φ of σ^{a} we say that the event is an

event of type φ. So, all events of type $a{:}move{=}l \wedge \neg\, c{:}move$ are also events of type $a{:}move{=}l$, and events of type $\neg\, c{:}move$, and so on. By extension, we also say that a transition is of type φ when its label (event) is of type φ. However, there are things we want to say about transitions that are not properties of their events (labels), in particular, whenever we want to refer to what holds in the initial state or final state of the transition. Transition formulas subsume event formulas but are more general. Although evaluating formulas on transitions seems to be unusual, representing events by Boolean compounds of propositional atoms is not so unusual. It is a feature of the action language $\mathcal{C}+$ [4], for example, and has also been used recently in [9] in discussions of agent 'ability'.

Formulas. Formulas are state formulas and transition formulas.

State formulas:

$$F \;::=\; \top \mid \bot \mid \text{any atom } f{=}v \text{ of } \sigma^{\mathrm{f}} \mid \neg F \mid F \wedge F \mid [\varphi]F$$

Transition formulas:

$$\varphi \;::=\; \top \mid \bot \mid \text{any atom } a{=}v \text{ of } \sigma^{\mathrm{a}} \mid \neg\varphi \mid \varphi \wedge \varphi \mid 0{:}F \mid 1{:}F$$

where F is any propositional state formula (i.e., a propositional formula of σ^{f}). We refer to the propositional formulas of σ^{a} as *event formulas.*

$\top$ and $\bot$ are 0-ary connectives with the usual interpretation. The other truth-functional connectives (disjunction $\vee$, material implication $\rightarrow$, and bi-implication $\leftrightarrow$) are introduced as abbreviations in the standard manner.

Models. Models are structures

$$\mathcal{M} = \langle S, A, R, h^{\mathrm{f}}, h^{\mathrm{a}} \rangle$$

where h^{f} and h^{a} are the valuation functions for state constants and event constants, respectively:

$$h^{\mathrm{f}}\colon S \rightarrow \mathrm{I}(\sigma^{\mathrm{f}}) \qquad \text{and} \qquad h^{\mathrm{a}}\colon A \rightarrow \mathrm{I}(\sigma^{\mathrm{a}})$$

$h^{\mathrm{f}}(s)$ is an interpretation of σ^{f}, i.e., a function which assigns to every constant f in σ^{f} a value v in $dom(f)$, and $h^{\mathrm{a}}(\varepsilon)$ is an interpretation of σ^{a}, i.e., a function which assigns to every constant a in σ^{a} a value v in $dom(a)$. Accordingly, for every state s in S and event/label ε in A we have:

$$\begin{aligned} \mathcal{M}, s \models f{=}v \quad &\text{iff} \quad h^{\mathrm{f}}(s)(f) = v \\ \mathcal{M}, \varepsilon \models a{=}v \quad &\text{iff} \quad h^{\mathrm{a}}(\varepsilon)(a) = v \end{aligned}$$

and for every transition τ in R:

$$\mathcal{M}, \tau \models a{=}v \quad \text{iff} \quad \mathcal{M}, label(\tau) \models a{=}v$$

It would be possible to introduce a third sort σ^R of propositional atoms for expressing properties of transitions, different from σ^{a} though not necessarily disjoint. A model would then include a third valuation function $h^R\colon R \to \mathrm{I}(\sigma^R)$ with

$$\mathcal{M}, t \models a{=}v \quad \text{iff} \quad h^R(\tau)(a) = v$$

We will not bother with that extension here. Event constants in σ^{a} are evaluated on both event/transition labels and transitions in the present set up. The difference is that event formulas are only the propositional formulas of σ^{a} whereas transition formulas are more general (as defined above). Transition formulas will be extended with some additional constructs in Sect. 6.

When φ is a formula of σ^{a} and τ is a transition in R we say that τ is a transition of type φ when τ satisfies φ, i.e., when $\mathcal{M}, \tau \models \varphi$, and sometimes that φ is true at, or true in, the transition τ. A state s satisfies a formula F when $\mathcal{M}, s \models F$. We sometimes say a formula F 'holds in' state s or 'is true in' state s as alternative ways of saying that s satisfies F.

Semantics. Let $\mathcal{M} = \langle S, A, R, h^{\mathrm{f}}, h^{\mathrm{a}}\rangle$ and let s and τ be a state and transition of $\mathcal{M}$ respectively. The satisfaction definitions for atomic propositions are described above. For negations, conjunctions, and all other truth functional connectives, we take the usual definitions. The satisfaction definitions for the other operators are as follows, for any state formula F and any transition formula φ.

State formulas:

$$\mathcal{M}, s \models [\varphi]F \quad \text{iff} \quad \mathcal{M}, \tau \models \varphi \text{ for every } \tau \in R \text{ such that } \mathit{prev}(\tau) = s.$$

$\langle\varphi\rangle$ is the dual of $[\varphi]$: $\langle\varphi\rangle F =_{\mathrm{def}} \neg[\varphi]\neg F$.

Transition formulas:

$$\begin{aligned} \mathcal{M}, \tau \models 0{:}F \quad &\text{iff} \quad \mathcal{M}, \mathit{prev}(\tau) \models F \\ \mathcal{M}, \tau \models 1{:}F \quad &\text{iff} \quad \mathcal{M}, \mathit{post}(\tau) \models F \end{aligned}$$

$$\|F\|^{\mathcal{M}} =_{\mathrm{def}} \{s \in S \mid \mathcal{M}, s \models F\}; \quad \|\varphi\|^{\mathcal{M}} =_{\mathrm{def}} \{\tau \in R \mid \mathcal{M}, \tau \models \varphi\}.$$

As usual, we say that F is *valid* in a model $\mathcal{M}$, written $\mathcal{M} \models F$, when $\mathcal{M}, s \models F$ for every state s in $\mathcal{M}$, and φ is *valid* in a model $\mathcal{M}$, written $\mathcal{M} \models \varphi$, when $\mathcal{M}, \tau \models \varphi$ for every transition τ in $\mathcal{M}$. A formula is *valid* if it is valid in every model $\mathcal{M}$ (written $\models F$ and $\models \varphi$, respectively).

$\mathcal{C}+$ [4] is a language for defining (a certain class of) transition systems of this type. The ICCALC implementation can be used to evaluate state, event, and transition formulas on transition systems defined by $\mathcal{C}+$ though it is not restricted to transition systems of that type.

Let us discuss the transition formulas first. A transition is of type $0{:}F$ when its initial state satisfies the state formula F, and of type $1{:}G$ when its resulting

state satisfies G. The following transition formula represents a transition from a state where (state atom) p holds to a state where it does not:

$$0{:}p \wedge 1{:}\neg p$$

von Wright [10] uses the notation $p\,\mathrm{T}\,q$ to represent a transition from a state where p holds to one where q holds. It would be expressed here as the transition formula:

$$0{:}p \wedge 1{:}q$$

Our notation is more general. We will make some further comments in Sect. 6.4.

For example, let the state atom $\mathit{on\text{-}table}(\mathit{vase})$ represent that a certain vase is standing on a table. A transition of type $0{:}\mathit{on\text{-}table}(\mathit{vase}) \wedge 1{:}\neg\,\mathit{on\text{-}table}(\mathit{vase})$, equivalently, of type $0{:}\mathit{on\text{-}table}(\mathit{vase}) \wedge \neg 1{:}\mathit{on\text{-}table}(\mathit{vase})$ is one from a state in which the vase is on the table to one in which it is not on the table. Suppose that the event atom $\mathit{falls}(\mathit{vase})$ represents the falling of the vase from the table. A vase-falling transition is also a transition from a state in which the vase is on the table to a state in which the vase is not on the table, and so any LTS model $\mathcal{M}$ modelling this domain will have the validity

$$\mathcal{M} \models \mathit{falls}(\mathit{vase}) \rightarrow (0{:}\mathit{on\text{-}table}(\mathit{vase}) \wedge 1{:}\neg\,\mathit{on\text{-}table}(\mathit{vase}))$$

There may be other ways that the vase can get from the table to the ground. Some agent might move the vase from the table to the ground, for example. That would also be a transition of type $0{:}\mathit{on\text{-}table}(\mathit{vase}) \wedge 1{:}\neg\,\mathit{on\text{-}table}(\mathit{vase})$ but not a transition of type $\mathit{falls}(\mathit{vase})$.

The operators 0: and 1: are both normal[3]. Since *prev* and *post* are (total) functions on R, we have

$$\models 0{:}F \leftrightarrow \neg 0{:}\neg F \qquad \text{and} \qquad \models 1{:}F \leftrightarrow \neg 1{:}\neg F$$

(which also means that 0: and 1: distribute over *all* truth-functional connectives).

Now some brief comments about state formulas. When φ is a transition formula, then $[\varphi]F$ is true at a state s when every transition of type φ from state s results in a state where F is true. $\langle\varphi\rangle F$ is true at a state s when there exists at least one transition of type φ from state s whose resulting state satisfies F. $[\varphi]\bot$, equivalently $\neg\langle\varphi\rangle\top$, says that there is no transition of type φ from the current state, and $\neg[\varphi]\bot$, equivalently $\langle\varphi\rangle\top$, that there is a transition of type φ from the current state. When α is an event formula, that is, a propositional formula of σ^{a}, then $\langle\alpha\rangle\top$, equivalently, $\neg[\alpha]\bot$ represents that an event of type α is executable in the current state.

It is important not to confuse the state formula $[\varphi]F$ with the notation $[\varepsilon]F$ used in Propositional Dynamic Logic (PDL). In PDL, the term ε in an expression $[\varepsilon]F$ is a transition label/event ε of A, not a transition *formula* as here. For

[3] This is standard terminology. See e.g. [11,12] or any introductory text on modal logic.

example, $[0{:}F \wedge \varphi]G$ and $\langle 0{:}F \wedge \varphi \wedge 1{:}G\rangle\top$ are both state formulas. The first is equivalent to $F \rightarrow [\varphi]G$ and the second to $F \wedge \langle\varphi\rangle G$.

The logic of each $[\varphi]$ is normal. Moreover:

$$\text{if } \mathcal{M} \models\ \varphi \rightarrow \varphi' \quad \text{then} \quad \mathcal{M} \models\ \langle\varphi\rangle F \rightarrow \langle\varphi'\rangle F$$

as is easily confirmed, and hence

$$\text{if } \mathcal{M} \models\ \varphi \rightarrow \varphi' \quad \text{then} \quad \mathcal{M} \models\ [\varphi']F \rightarrow [\varphi]F$$

We also have validity of:

$$([\varphi]F \wedge [\varphi']F) \rightarrow [\varphi \vee \varphi']F$$

and of

$$[\bot]\bot$$

Sauro et al. [9] have recently employed a similar device in a logic of agent 'ability' though in a more restricted form than we allow. (Their notation is slightly different.) They give a sound and complete axiomatisation for the logic of expressions $[\alpha]F$ where (in our terms) F is a propositional formula of σ^{f} and α is an event formula, that is, a propositional formula of σ^{a}. We will not present a complete axiomatisation of our more general language here. It is not essential for the purposes of this paper. We note only that an axiomatisation is more complicated for the more general expressions $[\varphi]F$ because there are some further relationships between state formulas and transition formulas that need to be taken into account. For example, all instances of the following state formulas are obviously valid

$$[1{:}F]F$$

as are all instances of

$$(F \rightarrow [\varphi]G) \ \leftrightarrow\ [0{:}F \wedge \varphi]G$$

Generally speaking, we find that properties of labelled transition systems are more easily and clearly expressed as transition formulas rather than state formulas. For example, although we cannot say using a transition formula that in a particular state of $\mathcal{M}$, every transition of type φ leads to a state which satisfies G, we can say (as we often want to) that whenever a state of $\mathcal{M}$ satisfies F, every transition of type φ from that state leads to a state which satisfies G. That is:

$$\mathcal{M} \models (0{:}F \wedge \varphi) \rightarrow 1{:}G$$

Properties of models can often be expressed equivalently as validities of state formulas or of transition formulas. This is because:

$$\mathcal{M} \models F \rightarrow [\varphi]G \quad \text{iff} \quad \mathcal{M} \models (0{:}F \wedge \varphi) \rightarrow 1{:}G$$

For example, suppose that the state atoms *light=on* and *light=off* represent the status of a particular light, and *loc*(x)=p that agent x is at location p.

Suppose that the (Boolean) event constant *toggle* represents that the light switch is toggled, and event constants $x{:}move$ with domain $\{l, r, \mathsf{f}\}$ that agent x moves in the direction l, r, or stays where it is. A model $\mathcal{M}$ modelling this domain would have the properties:

– State formulas

$$\mathcal{M} \models light{=}on \rightarrow [toggle]\, light{=}off$$
$$\mathcal{M} \models loc(x){=}p \rightarrow [\neg x{:}move]\, loc(x){=}p$$

– Transition formulas

$$\mathcal{M} \models (0{:}light{=}on \wedge toggle) \rightarrow 1{:}light{=}off$$
$$\mathcal{M} \models (0{:}loc(x){=}p \wedge \neg x{:}move) \rightarrow 1{:}loc(x){=}p$$

We find transition formulas are generally more useful and clearer.

2.2 Norms and Coloured Transition Systems

A simple way of representing norms is to partition the states and transitions of a transition system into two categories. A *coloured transition system* [5,6] is a structure of the form $\langle S, A, R, S_g, R_g\rangle$ where $\langle S, A, R\rangle$ is a labelled transition system of the kind discussed above, and where the two new components are

– $S_g \subseteq S$, the set of 'permitted' ('acceptable', 'ideal', 'legal') states—we call S_g the 'green' states of the system;
– $R_g \subseteq R$, the set of 'permitted' ('acceptable', 'ideal', 'legal') transitions—we call R_g the 'green' transitions of the system.

We refer to the complements $S_{red} = S \setminus S_g$ and $R_{red} = R \setminus R_g$ as the 'red states' and 'red transitions', respectively. Semantical devices which partition states (and here, transitions) into two categories are familiar in the field of deontic logic. For example, Carmo and Jones [13] employ a structure which has both ideal/sub-ideal states and ideal/sub-ideal transitions (unlabelled). van der Meyden's 'Dynamic logic of permission' [14] employs a structure in which transitions, but not states, are classified as 'permitted/non-permitted'. van der Meyden's version was constructed as a response to problems of Meyer's 'Dynamic deontic logic' [15] which classifies transitions as 'permitted/non-permitted' by reference to the state resulting from a transition. 'Deontic interpreted systems' [16] classify states as 'green'/'red', where these states have further internal structure to model the local states of agents in a multi-agent context. Recently, Ågotnes et al. [17] have presented a language based on the temporal logic CTL. They partition transitions into those that comply with a set of norms and those that do not (that is, into 'green' and 'red' in our terminology). They then define a modified form of CTL for expressing temporal properties of paths/runs in which every transition is 'green', or what we refer to as 'fully compliant behaviour' in Sect. 4.1 below. There are no constructs in the language for expressing properties of paths/runs in which some transition is not 'green'.

We require that a coloured transition system $\langle S, A, R, S_g, R_g \rangle$ must further satisfy the constraint that, for all states s and s' in S and all transitions τ in R:

$$\text{if } \tau \in R_g \text{ and } \mathit{prev}(\tau) \in S_g \text{ then } \mathit{post}(\tau) \in S_g \tag{1}$$

We refer to this as the *green-green-green* constraint, or *ggg* for short. (It is difficult to find a suitable mnemonic.)

The *ggg* constraint (1) expresses a kind of *well-formedness* principle: a green (permitted, acceptable, legal) transition in a green (permitted, acceptable, legal) state always leads to a green (acceptable, legal, permitted) state. It may be written equivalently as:

$$\text{if } \mathit{prev}(\tau) \in S_g \text{ and } \mathit{post}(\tau) \in S_{red} \text{ then } \tau \in R_{red} \tag{2}$$

Any transition from a green (acceptable, permitted) state to a red (unacceptable, non-permitted) state must itself be undesirable (unacceptable, non-permitted), i.e., 'red', in a well-formed system specification.

One can consider a range of other properties that we might require of a coloured transition system: for example, that the transition relation must be serial (i.e., that there is at least one transition from every state), or that there must be at least one green state, or that from every green state there must be at least one green transition, or that from every green state reachable from some specified initial state(s) there must be at least one green transition, and so on. These are examples of properties that might be of interest when analyzing a transition system. We can check for them but we do not assume they are always satisfied. We do assume that every coloured transition systems satisfies the *ggg* constraint.

Instead of introducing a special category of coloured transition systems, with extra components S_g and R_g, we now prefer to speak of labelled transition systems generally and introduce colourings for states and transitions by means of suitably chosen constants in σ^{f} and σ^{a}. This is more general and adds flexibility. In particular, we have a state constant *status* and an event constant *trans* both with domain $\{\mathit{green}, \mathit{red}\}$. The intended reading is that $\|\mathit{status}{=}\mathit{green}\|^{\mathcal{M}}$ denotes the 'green states' and $\|\mathit{status}{=}\mathit{red}\|^{\mathcal{M}} = S \setminus \|\mathit{status}{=}\mathit{green}\|^{\mathcal{M}}$ the 'red states'; $\|\mathit{trans}{=}\mathit{green}\|^{\mathcal{M}}$ denotes the 'green transitions' and $\|\mathit{trans}{=}\mathit{red}\|^{\mathcal{M}} = R \setminus \|\mathit{trans}{=}\mathit{green}\|^{\mathcal{M}}$ the 'red transitions'.

The *ggg* constraint (1) can then be expressed as validity in any model $\mathcal{M}$ of the state formula

$$\mathit{status}{=}\mathit{green} \rightarrow [\mathit{trans}{=}\mathit{green}]\,\mathit{status}{=}\mathit{green}$$

or, equivalently, of the transition formula

$$(0{:}\mathit{status}{=}\mathit{green} \wedge \mathit{trans}{=}\mathit{green}) \rightarrow 1{:}\mathit{status}{=}\mathit{green}$$

As further illustrations of the use of the language, here are the other properties mentioned earlier, expressed now as validities in a model $\mathcal{M}$.

– The transition relation must be serial

$$\mathcal{M} \models \langle \top \rangle \top$$

– There must be at least one green state

$$\mathcal{M} \not\models \mathit{status}{=}\mathit{red}, \quad \text{equivalently, } \mathcal{M} \not\models \neg(\mathit{status}{=}\mathit{green})$$

– From every green state there must be at least one green transition

$$\mathcal{M} \models \mathit{status}{=}\mathit{green} \rightarrow \langle \mathit{trans}{=}\mathit{green} \rangle \top$$

We cannot express, in this language, that from every green state reachable from some specified initial state(s) there must be at least one green transition since we have no way of expressing reachability (in the language). That could be fixed by extending the language but we will not do it here. Reachability properties in a model can be checked using the ICCALC system but are not expressible as formulas of the language.

$n\mathcal{C}+$ [5,6] is a language for defining (a certain class of) transition systems of this type. The ICCALC implementation builds in the special treatment of 'red' and 'green' required to ensure that the *ggg* constraint is satisfied.

In [6] we presented a refinement where instead of the binary classification of states as red or green, states are ordered according to how well each complies with the state permission laws of an $n\mathcal{C}+$ action description. We also discussed possible generalisations of the *ggg* constraint for that case. In the current paper, we keep to the simple classification of states as green or red.

Notice that we would get much more precision by colouring *paths/runs* of the transition system instead of just its states and transitions. One could then extend the logics presented in this paper with features from a temporal logic such as CTL. The details seem straightforward but we leave them for future investigation.

3 Example (Rooms)

This example concerns the specification of norm-governed interactions between independently acting agents. It was discussed in a previous paper [1]. We now present it using the formalism introduced in previous sections.

In the example there are two categories of agents, male and female, who move around in a world of interconnecting rooms. The rooms are connected by doorways through which agents may pass. (The precise topography, and number of rooms, can vary.) Each doorway connects two rooms. Rooms can contain any number of male and female agents. The action atoms of σ^{a} will take the form $x{:}\mathit{move}{=}p$, where x ranges over the agents in a particular example, and p ranges over a number of values representing directions in which agents can move, in addition to a value f: if a transition satisfies $x{:}\mathit{move}{=}\mathsf{f}$, that is to be taken to represent that agent x does not move during that transition. Recall that by convention we write $\neg x{:}\mathit{move}$ as a shorthand for $x{:}\mathit{move}{=}\mathsf{f}$.

A normative element is introduced by insisting that a male agent and a female agent may never be alone together in a room; such configurations are physically possible, and the transition system will include states representing them, but all such states will be coloured red.

Although this example is relatively simple, it shares essential features with a number of real-world domains, in which there are large numbers of interacting agents or components which may be in different states, and where some of those combinations of states are prohibited. (These real-world examples are *not* restricted to domains where agents perform physical actions. Exactly the same points could be made for examples of institutions or virtual organisations, where the possible actions by agents are defined and constrained by institutional rules rather than physical constraints, and where actions by agents can be represented as transitions from one institutional state to another.)

For the purposes of illustration, we shall consider a concrete instance of the example in which there are just two rooms, on the left and right, with one connecting door, and three agents, two males m_1 and m_2, and a female f_1. We have deliberately made the example simple in order to concentrate on its essential features, and so that we can depict the transition system in its entirety. With more agents and more rooms the transition system is too big to be shown easily in diagrammatic form. We will also impose an additional constraint that only one agent can move through the doorway at once (the doorways are too narrow to let more than one agent pass through at the same time). This is a more significant restriction since it imposes constraints on possible interactions between the agents: if an agent moves from one room to another it thereby makes it impossible for other agents to pass through the same doorway.

The propositional language for this instance of the 'rooms' example contains state atoms $loc(x)$=l and $loc(x)$=r, where x ranges over m_1, m_2, f_1; $loc(m_1)$=l is true when the male agent m_1 is in the left-hand room, $loc(m_2)$=r is true when m_2 is in the right-hand room, and so on. The action atoms are, in line with previous remarks, x:$move$=p, where x ranges over the agents and p ranges over $l, r, \mathtt{f}$.

We do not show the $n\mathcal{C}+$ formulation of the example here. (It can be found in [1].) The transition system, whether defined using $n\mathcal{C}+$ or by some other means, is depicted in Fig. 1. We have not drawn the transitions from states to themselves, where no agent moves, in order to keep the drawing clear; all such transitions are coloured green. Also, we have not shown labels for transitions. These can easily be deduced, for every arc in the diagram should have a label which makes precisely one x:$move$=p atom true (for p one of $l, r, \mathtt{f}$); for example, the (red) transition from the top-most state (`s1`) to the one immediately below and to the right (`s6`) has a label which makes m_2:$move$=r, $\neg m_1$:$move$, and $\neg f_1$:$move$ true.

One can see that the transition system satisfies the *ggg* constraint: since nothing further was said about the colouring of transitions, the red transitions are simply those where the system moves from a green state to a red state, i.e., to a state in which a male and a female are alone in a room together.

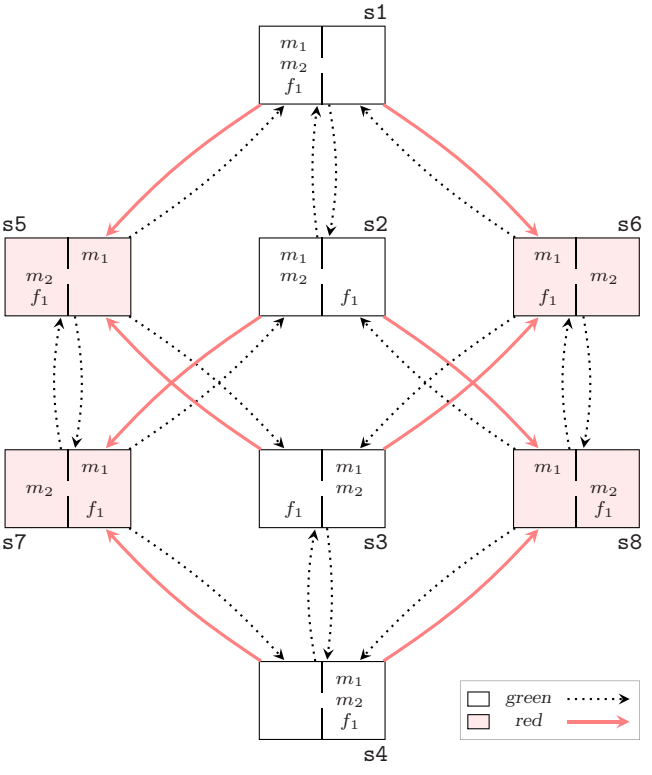


Fig. 1. A simple 'rooms' example

However, the example is also intended to demonstrate some important inadequacies. For consider, again, the transition from the top-most state (`s1`) where all agents are in the left-hand room, to the state below and to the right of it (`s6`) where m_1 and f_1 are left alone together after m_2 has exited to the right. In some sense, it is m_2 who has acted wrongly: m_2 has left the room, leaving m_1 and f_1 alone together, in a configuration which thus violates the norms governing the system. On the other hand, if we remove the restriction that at most one agent can pass through the doorway at one time, it is far from clear which of the three agents, if any, acted wrongly when m_2 exited: it might have been m_2 who acted badly, or it might have been m_1, who should have followed m_2 out, or it might have been f_1, who should have followed m_2 out. Or it might be that all of them, collectively, acted wrongly, or perhaps none of them. The transition systems as they currently stand do not have the capacity to represent that it is specifically one agent's actions rather than another's which must be marked as 'red'. There is no way to extract from, or represent in, the transition system that a particular agent's actions in the transition are illegal, sub-ideal, undesirable, and so on; indeed, there is no explicit concept of an individual agent in the semantics at all.

4 Agent-Specific Norms

The language of the previous sections provides a means of representing when states and transitions satisfy, or fail to satisfy, a standard of legality, acceptability, desirability, and so on. Much can be said using the resources of this language. However, in representing systems in which there are multiple interacting agents (as with the simple 'rooms' example depicted in the previous section), it is often essential to be able to speak about an individual agent's behaviour: in particular, about whether individual agents' actions are in the right or wrong—whether they are conforming to norms which govern specifically *their* behaviour. In [1] we introduced a semantical structure which we called a *coloured agent-stranded transition system*. That had two components: a way of picking out an individual agent x's actions from a transition—the agent x's 'strand' in that transition, and a colouring of each such strand as $green(x)$ or $red(x)$ to represent the agent-specific norms for x. We will deal first with the agent-specific colourings $green(x)$ and $red(x)$ and defer discussion of the 'strand' component until Sect. 5.

In the context of using norms or 'social laws' to regulate the interactions of multiple, independently acting agents in a multi-agent computer system, the colourings of states and transitions as 'green' or 'red' represent *system norms*. They express a system designer's point of view of what system states and transitions are legal, permitted, desirable, and so on. There is a separate category of individual *agent-specific* norms that are intended to guide an individual agent's behaviours and are supposed to be taken into account in the agent's implementation, or reasoning processes, in one way or another. These have a different character. In order to be effective, or even meaningful, they must be formulated in terms of what an agent can actually sense or perceive and the actions that it can actually perform. So, in the 'rooms' example an agent-specific norm could not meaningfully prohibit an agent from acting in such a way that a male and female are alone in a room together. The agent cannot predict how other agents will act: just because a room is currently vacant, for example, does not mean that another agent will not enter it.

We now extend the 'rooms' example with some agent-specific norms. As a concrete example (one of many that could be devised) let us attempt to specify an (imperfect) protocol for recovery from red system states: whenever a male agent and a female agent are alone in a room, anywhere, every male agent is required to move to the room to its left (if there is one), and every female agent is required to move to the room to its right (if there is one).

Let Ag be a finite set of agent names. In the present example Ag=$\{m_1, m_2, f_1\}$. For each agent $x \in Ag$, $green(x)$ is a subset of R representing those transitions where the actions of x have been in accordance with norms specific for x. $red(x) = R \setminus green(x)$ are those transitions in which the actions of x have failed to conform to x's norms.

So in the example: suppose s is a state of the system in which there is a male agent and a female agent alone in a room. For every male agent x (anywhere), a transition from s in which x moves to the room on its left is coloured $green(x)$, a transition from s in which x stays where it is when there is no room to its left

is *green*(x), and any other transition from s is *red*(x). And similarly for female agents, but with 'left' replaced by 'right'. Further (let us suppose) in a state s of the system where there is not a male agent and a female agent alone in a room, for any agent x, any transition from s is *green*(x). Thus, the agents are free to move around from room to room, but if ever the system enters a red global state, their individual norms require them to move to the left or right as the case may be; once the system re-enters a green global state they are free to move around again.

The precise mechanism by which agents detect that there is a male agent and a female agent alone in a room somewhere is not modelled at this level of detail. We will simply assume that there is some such mechanism—a klaxon sounds, or a suitable message is broadcast to all agents—the details do not matter for present purposes. Similarly, we are not modelling here how an agent determines which way to move. In a more detailed representation, we could model an agent's internal state, its perceptions of the environment in which it operates, how it determines where to move, and the mechanism by which it perceives that there is a male agent and a female agent alone in a room. We will not do so here: the simpler model is sufficient for present purposes.

In general, given a transition system modelling all the possible system behaviours, and some (finite) set Ag of agent names, we specify for every agent x in Ag the norms specific to x that govern x's individual actions: some subset of the transitions in a given system state will be designated as *green*(x) and the others as *red*(x). In the example as we have it, the agent-specific norms only constrain the agents' actions in a red system state. That is not essential. It is merely a feature of this particular example. A transition is designated as *green*(x) when x's actions in that transition comply with the agent-specific norms for x. We specify, separately, system norms which constrain various combinations of actions by individual agents, or other interactions of interest, by classifying transitions and states as globally red or green. So we have two separate layers of specification: (i) norms specific to agents governing their individual actions, and (ii) norms governing system behaviour as a whole. We are interested in examining the relationships, if any, between these two separate layers. We might be interested in verifying, for example, that all behaviour by agent x compliant with the norms for x guarantees that the system avoids globally red states, or produces only globally green runs, or always recovers from a global red state to a global green state, and so on. This is the setting we have in mind for discussion in this paper. We also want to identify several different categories of non-compliant behaviours, and generally, the conditions under which we can say that it is a particular agent x's actions that are responsible for, or the cause of, a transition being coloured (globally) red, or more generally, being of type φ.

As in the case of coloured transition systems discussed earlier, we prefer to speak of transition systems in general, and use suitably chosen event constants to represent the properties of interest. So, let σ^{a} contain (Boolean) event constants *green*(x) for every agent $x \in Ag$, and let *red*(x) be an abbreviation for $\neg$ *green*(x). A transition τ in R is, or is coloured, *green*(x), respectively *red*(x), in a model $\mathcal{M}$ when $\mathcal{M}, \tau \models$ *green*(x), or $\mathcal{M}, \tau \models$ *red*(x), respectively. The *green*(x) transitions

in a model $\mathcal{M}$ are $\|green(x)\|^{\mathcal{M}}$; the $red(x)$ transitions are $\|red(x)\|^{\mathcal{M}} = R \setminus \|green(x)\|^{\mathcal{M}}$.

We retain the *ggg* constraint for the colouring of states and transitions as (globally) green or red as determined by the system norms. There is no analogue of the *ggg* constraint for the colourings representing agent-specific norms. However, it is natural to consider an optional *coherence constraint* relating the agent-specific colourings of a transition to its global (system norm) colouring. The colouring of a transition as (globally) red represents that the system as a whole fails to satisfy the required standard of acceptability, legality, desirability represented by the global green/red colouring. In many settings it is then natural to say that if any one of the system components (agents) fails to satisfy its standards of acceptability, legality, desirability, then so does the system as a whole: if a transition is $red(x)$ for some agent x then it is also (globally) red. Formally, the model $\mathcal{M} = \langle S, A, R, h^{\mathrm{f}}, h^{\mathrm{a}} \rangle$ satisfies the local-global coherence constraint whenever, for all agents $x \in Ag$, $red(x) \subseteq R_{red}$, that is to say, when

$$\mathcal{M} \models red(x) \rightarrow trans{=}red \tag{3}$$

The coherence constraint (3) is optional and not appropriate in all settings. We will adopt it in the examples discussed below. Notice though, that even if the coherence constraint is adopted, it is possible that a transition can be coloured $green(x)$ for every agent x and still itself be coloured globally red. We will give some examples presently.

There are other, more fundamental constraints that we must place on agent-specific colourings. We defer discussion of those until Sect. 5.

4.1 Fully Compliant Behaviour

As suggested above, we might now be interested in examining the relationship between system norms and individual agent-specific norms—in the present example, for instance, to determine whether the agent-specific norms expressed by the $green(x)$ specification do have the desired effect of guaranteeing recovery from a red system state to a green system state. Given a coloured transition system representing the system norms and the agent-specific norms, defined by an $n\mathcal{C}+$ action description or by some other means, we focus attention on those paths of the transition system that start at a red system state, and along which every agent always acts in accordance with its norms, i.e., those paths in which every transition is $green(x)$ for each of the agents x. A natural property to look for is whether all such paths eventually pass through a green system state; if this property holds, it indicates that the agent-specific norms are doing a good job in ensuring that systems in violation of their global system norms eventually recover to a green state, assuming that all agents follow their individual norms correctly. (There is a further natural requirement: in the case where the system is initially in a red system state s, there should be at least one transition from that state. Otherwise, the requirement that all paths starting at s eventually reach a green system state would be vacuously satisfied.)

In particular applications, it might not be a reasonable assumption to make that agents always act in accordance with their individual norms. This might be for several reasons. Sometimes physical constraints in the environment being modelled prevent joint actions in which all agents act well; in other circumstances, and noteworthy especially because we have in mind application areas in multi-agent systems, agents may not comply with the norms that govern them because it is more in their interests not to comply. In the latter case, penalties are often introduced to try and coerce agents into compliance. We leave that discussion to one side, however, as it is tangential to the current line of enquiry.

Consider now the 'rooms' example in particular, and what happens if we assume that all agents act in accordance with their individual norms. It is clear that the effectiveness of the protocol (if in a red state, males move to the left when possible, females move to the right when possible) in guaranteeing that the system will eventually reach a green state, depends on the topography of rooms and connecting doors. Let us assume that there is a finite number of rooms, each room has at least one connecting room to its left or one to its right, and that there are no cycles in the configuration, in the sense that if an agent continues moving in the same direction it will never pass first out of, then back into, the same room. Under these circumstances, and removing the restriction on how many agents can pass through a door at the same time, it is intuitive that there is always a recovery, in the sense defined, from every red system state. Since all agents act in accordance with their norms, every male will move to the left (if it can), and every female will move to the right (if it can). If the resulting system state is not green, they will move again. Eventually, in the worst case, the males and females will be segregated in separate rooms, which is a green system state.

Of course, we cannot guarantee that having reached a green system state, the agents will not re-enter a red state: in this example, the individual agent-specific norms only dictate how agents should behave when the system is globally red. Once the system has recovered, the agents may mingle again. It is easy to imagine how we might use a model-checker to verify this and similar properties on coloured transition systems; we will not discuss the details in this paper.

4.2 Non-compliant Behaviours

One must be careful not to assume that if an agent x fails to comply with its individual norms—if some transition τ is $red(x)$—then it must be that agent x acted wilfully, perhaps to seek some competitive advantage, or carelessly; or if it is a simple reactive device, that its constructors failed to implement it correctly. This may be so, but an agent may also fail to comply with its norms because of factors beyond its control, because it is prevented from complying by the actions of other agents, or by extraneous factors in the environment. To illustrate: suppose we return to the version of the 'rooms' example in which it is impossible for more than one agent to pass through the same doorway at the same time. All other features, including the specification of system norms and agent-specific norms, remain as before. Clearly the situation can now arise where several agents are required by their individual norms to pass through the same

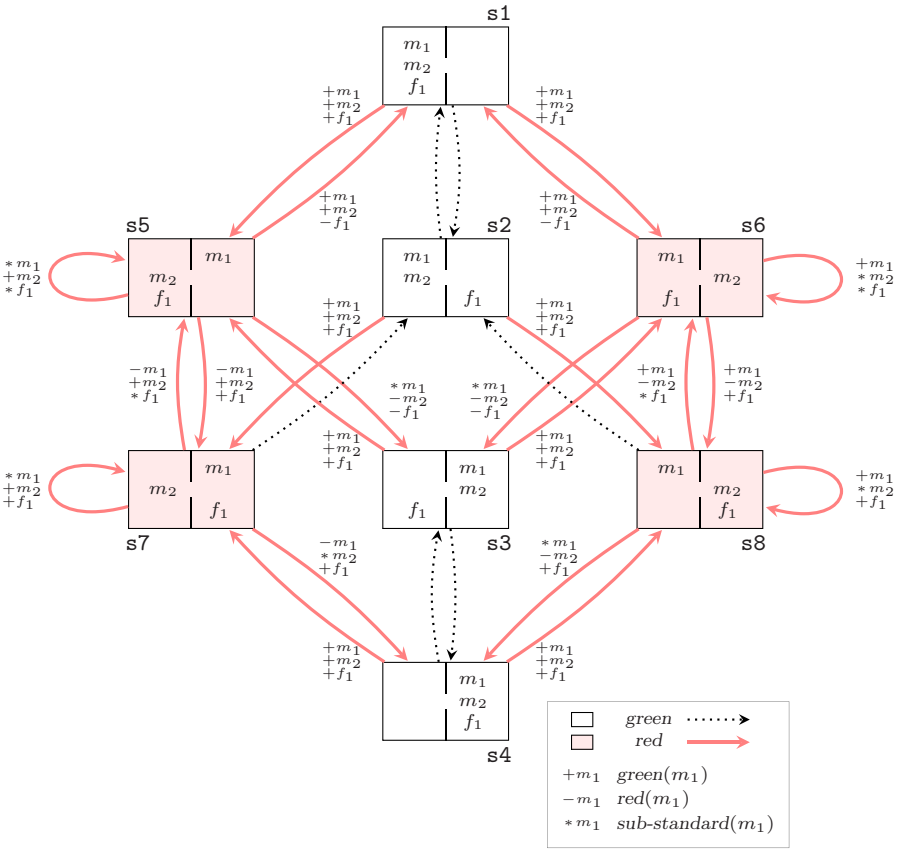


Fig. 2. Transitions without annotations are *green*(x) for each of the three agents x. Reflexive arcs on green nodes, where no agent moves, are omitted from the diagram: they are all globally green, and *green*(x) for each agent x. (The concept of a *sub-standard* strand is explained in Sect. 4.3.)

doorway; at most one of them can comply, and if one does comply, the others must fail to comply.

Again, in order to keep diagrams of the transition system small enough to be shown in full, we will consider just the case of two interconnecting rooms, and three agents, m_1, m_2, and f_1, of whom the first two are male and the last is female. Figure 2 shows the coloured agent-stranded transition system for this version of the example. We have adopted here the local-global coherence constraint (3) which is why some transitions that were globally green in the version of Sect. 3 are now globally red. Nothing essential in what follows depends on this. Transition labels are omitted from the diagram: since at most one agent can move at a time, they are obvious from looking at the states. Annotations on the arcs indicate the three agent-specific colourings for each transition; where arcs have no such annotation the transition is *green*(x) for each of the three agents

x. Omitted from the diagram are reflexive arcs from the green system states to themselves, representing transitions in which no agent moves. These transitions are all globally green, and therefore also (given local-global coherence) *green*(x) for each agent x. The significance of the asterisks in some of the annotations will be explained presently. For now they may be read as indicating that the transition is *red*(x) for the agent x.

One can see from the diagram that the system exhibits the following kinds of behaviour, among others.

(1) There are transitions coloured *green*(x) for all three agents x but which are nevertheless globally red. This is because, in this example, the agent-specific norms do not constrain agents' actions in green system states. Indeed, one can see from the diagram that in this example (though not in general) the globally red transitions which are *green*(x) for all three agents x are exactly those from a green system state to a red system state. The model $\mathcal{M}$ has the property:

$$\mathcal{M} \models \mathit{green}(m_1) \wedge \mathit{green}(m_2) \wedge \mathit{green}(f_1) \wedge \mathit{trans{=}red} \leftrightarrow 0{:}\mathit{status{=}green} \wedge 1{:}\mathit{status{=}red}$$

(2) There are globally green transitions from red system states to green system states (such as the one from state `s8` to state `s2` in which m_2 moves to the left and m_1 and f_1 stay where they are). These are transitions in which all three agents are able to comply with their individual norms. In this example, though not necessarily in other versions with more elaborate room configurations and more agents, such transitions always recover from a red system state to a green system state. The system exhibits the property:

$$\mathcal{M} \models \mathit{green}(m_1) \wedge \mathit{green}(m_2) \wedge \mathit{green}(f_1) \wedge 0{:}\mathit{status{=}red} \rightarrow 1{:}\mathit{status{=}green}$$

(3) There are also globally red transitions in which at least one agent fails to comply with its individual norms but which lead from a red system state to a green system state (such as the one from state `s8` to state `s4` in which m_1 moves to the right and m_2 and f_1 stay where they are). These transitions recover from a red system state to a green system state but in violation of the individual agent-specific norms. These are transitions of type

$$\mathit{trans{=}red} \wedge (\mathit{red}(m_1) \vee \mathit{red}(m_2) \vee \mathit{red}(f_1)) \wedge 0{:}\mathit{status{=}red} \wedge 1{:}\mathit{status{=}green}$$

In fact, in the rooms example, though not in general, the system has the property:

$$\mathcal{M} \models \mathit{trans{=}red} \wedge 0{:}\mathit{status{=}red} \wedge 1{:}\mathit{status{=}green} \rightarrow (\mathit{red}(m_1) \vee \mathit{red}(m_2) \vee \mathit{red}(f_1))$$

(4) There are globally red transitions, such as the one from state `s6` to state `s3` in which m_1 moves to the right, and f_1 and m_2 stay where they are, in which no agent complies with its individual norms. These are transitions of type

$$\mathit{trans{=}red} \wedge \mathit{red}(m_1) \wedge \mathit{red}(m_2) \wedge \mathit{red}(f_1)$$

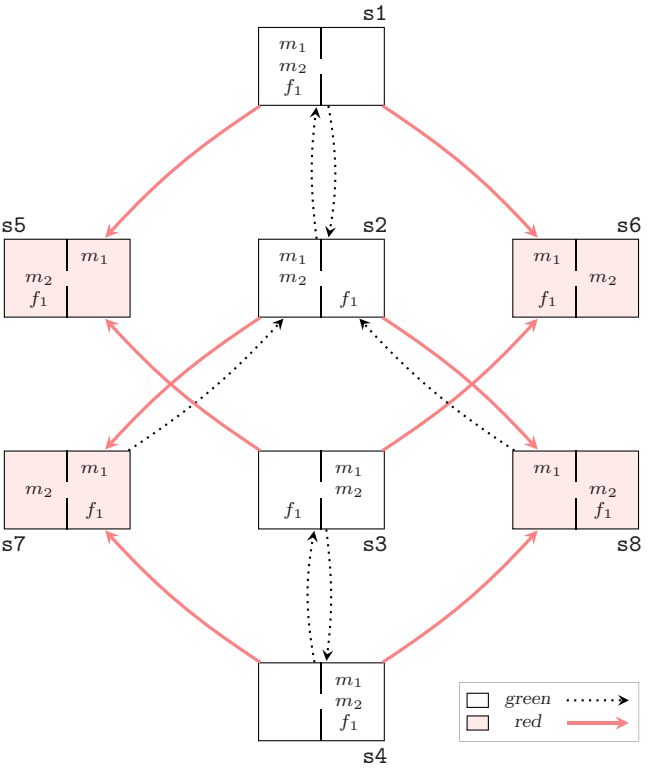


Fig. 3. System behaviour if all three agents comply with their individual norms. Reflexive arcs on green nodes are omitted from the diagram.

(5) And as the example is designed to demonstrate, there are globally red transitions where one agent complies with its individual norms but in doing so makes it impossible for one or both of the others to comply with theirs. For example, in the red system state `s6` where m_1 and f_1 are in the room on the left and m_2 is on the room on the right, there is no transition in which m_2 and f_1 can both comply with their individual norms: the following state formula is true at `s6`

$$\neg\langle \mathit{green}(m_2) \wedge \mathit{green}(f_1)\rangle \top \quad \text{equivalently} \quad [\mathit{green}(m_2) \wedge \mathit{green}(f_1)]\bot$$

$\mathit{green}(m_2) \wedge \mathit{green}(f_1)$ is not 'executable' in state `s6`.

In this version of the example, what are the possible system behaviours in the case where all agents do comply with their individual norms? Figure 3 shows the transition system that results if all $\mathit{red}(x)$ transitions are discarded, for all three agents x. The diagram confirms that when there is a constraint preventing more than one agent from moving through a doorway at a time, the system can enter a state from which there is no transition unless at least one agent fails to comply with its individual norms. In the diagram, these are the two red system

states s5 and s6 where the female agent f_1 is in the left-hand room with a male. The ICCALC system provides facilities for undertaking this kind of analysis.

Now: one may think that there is a flaw in the way that the individual agent-specific norms in the example have been formulated, that their specification is wrong in that there are situations which make norm compliance impossible. A properly designed set of norms, it might be argued, must satisfy an 'ought implies can' principle; if it does not, it is flawed. That is not so. We are thinking here of a multi-agent system in which agents act independently, where there is no communication between agents, and where no agent can predict how other agents will act. If there were such communication it might be different, but suppose there is not. In these circumstances, it is quite impractical to try to anticipate every possible combination of behaviours by other agents, and in the environment, and to try to formulate agent-specific norms that make provision for each eventuality. It is quite impractical, even in examples as simple as this. It is realistic, however, to formulate agent-specific norms that will guide an individual agent's behaviour without reference to what other agents might do, and simply accept that there might be circumstances in which the agent-specific norms for x conflict with those for y, and generally, that an agent may be prevented from complying with its individual agent-specific norms in some circumstances.

4.3 Sub-standard Behaviours

The example is designed to demonstrate several different categories of non-compliant agent behaviour. We pick out one for particular attention. Consider the state in which m_1 and f_1 are in the room on the left and m_2 is in the room on the right. (This is the red system state s6 at the upper right of the diagram.) Because of the constraint on moving through the doorway, it is not possible for all three agents to comply with their individual norms. But suppose that each agent behaves in such a way that it will comply with its individual norms *in as much as it can*. A purely reactive agent, let us suppose, is programmed in such a way that it will attempt to act in accordance with its individual norms though it may not always succeed if something prevents it. A deliberative agent (human or computer) incorporates its individual norms in its decision-making procedures and takes them into account when planning its actions: it will always attempt to act in accordance with its individual norms though it may be unsuccessful. If all agents in the system behave in this way, then there are two possible transitions from the red system state s6: either f_1 succeeds in moving to the right in accordance with its individual norms, or m_2 succeeds in moving to the left in accordance with its. The third possible transition from this system state, in which every agent stays where it is, can be ignored: it can only occur if no agent attempts to act in accordance with its individual norms, and this, we are supposing, is not how the agents behave. The exact mechanism which determines which of the two agents m_2 and f_1 is successful in getting through the doorway is not represented at the level of detail modelled here. At this level of detail, all we can say is that one or other of the agents m_2 and f_1 will pass through the doorway but we cannot say which.

Similarly, in the red system state `s8` at the lower right of the diagram, in which m_1 is on the left and m_2 and f_1 are on the right, we can ignore the transition in which m_1 moves to the right, if m_1's behaviour is such that it always attempts to comply with its individual norms. The transition in which f_1 moves to the left can also be ignored, if f_1's behaviour is to attempt to comply with its individual norms. And the transition in which m_2 stays where it is can be ignored, if m_2's behaviour is to attempt to comply with its individual norms. This leaves just one possible transition, in which m_2 attempts to move to the left; this will succeed because the other two agents will not act in such a way as to prevent it. (We are tempted to refer to this kind of behaviour as behaviour in which every agent 'does the best that it can'. The term has too many unintended connotations, however, and so we avoid it.)

We are not suggesting, of course, that agents *always* behave in this way, only that there are circumstances where they do, or where it can be reasonably assumed that they do, or simply where we are interested in examining what system behaviours result if we suppose that they do.

We now make these ideas a little more precise. We will say that x's behaviour in a particular transition τ from a state s is *sub-standard*(x) if the transition is *red*(x) and, had x acted differently in state s while all other agents acted in the same way as they did in τ, the transition from state s could have been *green*(x): x could have acted differently in state s and complied with its individual norms.

Alternatively, as another way of looking at it, we could say that a *red*(x) transition τ from a state s is *unavoidably-red*(x) if every transition from state s in which every agent other than x acts in the same way as it does in τ is also *red*(x): there is no *green*(x) transition from state s if every agent other than x acts in the way it does in τ. This is closer to the informal discussion above. One can see, informally for now, that every *red*(x) transition is *sub-standard*(x) if and only if it is not *unavoidably-red*(x), and indeed, that every *red*(x) transition is either *sub-standard*(x) or *unavoidably-red*(x), but not both.

Notice that these definitions allow for the possibility of actions in the environment. It is easy to imagine other versions of the example where an agent may be unable to act in accordance with its individual norms not because of the actions of other agents but because of extraneous factors in the environment. (Suppose, for instance, that an agent is unable to move to the room on the left while it is raining.) And here is a reason why we prefer not to treat 'the environment' as a kind of agent: we do not want to be talking about *sub-standard* behaviours of the environment, or of agents preventing the environment from acting in accordance with its individual norms. In this respect at least, 'the environment' is a very different kind of agent from the others.

It still remains to formalise these definitions. For this we need to be able to refer to actions by individual agents in transitions, which is not part of the LTS structure as we have it. Indeed, there is no explicit concept of an individual agent in the semantics at all. We defer further discussion until the next section. For now, we rely on the informal account just given.

The diagram of the transition system for this example was shown earlier in Fig. 2. The figure shows the sub-standard transitions for each agent. They are those in which the transition annotations are marked with an asterisk. For example, in the red system state `s6` at the upper right of the diagram, where m_1 and f_1 are on the left and m_2 is on the right, the transition in which all three agents stay where they are is *sub-standard*(m_2), because there is a *green*(m_2) transition from this state in which m_1 and f_1 act in the same way and m_2 acts differently, namely the transition in which m_1 and f_1 stay where they are and m_2 moves to the left in accordance with its individual norms. Similarly, the transition from state `s6` in which m_1 moves to the right and m_2 and f_1 stay where they are is *sub-standard*(m_1) because the transition where all three agents stay where they are is *green*(m_1). And likewise for the other transitions marked as sub-standard in the diagram. The *red*(x) transitions not marked as *sub-standard*(x) are *unavoidably-red*(x).

Suppose we wish to examine what system behaviours result if all three agents comply, in as much as they can, with their individual norms. Suppose, in other words, that we disregard those transitions which are sub-standard for any of the three agents x. The ICCALC implementation supports this kind of analysis. The result is shown in Fig. 4. There are still *red* transitions in the diagram. Some, such as the one from `s4` to `s8`, are *green*(x) for every x but are nevertheless globally red. Those, such as the one from `s6` to `s8`, which are *red*(x) for some agent x are *unavoidably-red*(x).

Many other variations of the example could be examined in similar fashion. If female agents are more reliable than male agents, for instance, we might be interested in examining what system behaviours result when there is never sub-standard behaviour by females though possible sub-standard behaviour by males.

Interestingly, when analysing the example using ICCALC, it turned out that if we assume there is no sub-standard behaviour by either of the two male agents m_1 and m_2, that is, if we assume that m_1 and m_2 always comply with their individual norms if they can, then there is no sub-standard behaviour by the female agent f_1 either. This is really an artefact of the simplicity of the example where there are just two rooms and very strong constraints on how the three agents can move between them. Nevertheless, it does demonstrate the possibility, in principle at least, that agents can sometimes be coerced into compliance by the behaviours of others, without resort to sanctions and other enforcement mechanisms.

As a final remark, notice that what is *sub-standard*(x) or *unavoidably-red*(x) for an agent x can depend on *normative* as well as *physical* constraints. Suppose (just for the sake of an example) that there is another individual norm for m_1 to the effect that it should never stay in a particular room (say, the room on the left) but should move out immediately if it enters it: a transition in which m_1 stays in the room on the left is *red*(m_1), in every system state, red or green. With this additional constraint, some of the transitions that were globally green are now globally red because of the local-global coherence constraint (assuming we choose to adopt it). But further, the transition from the red system state `s6`

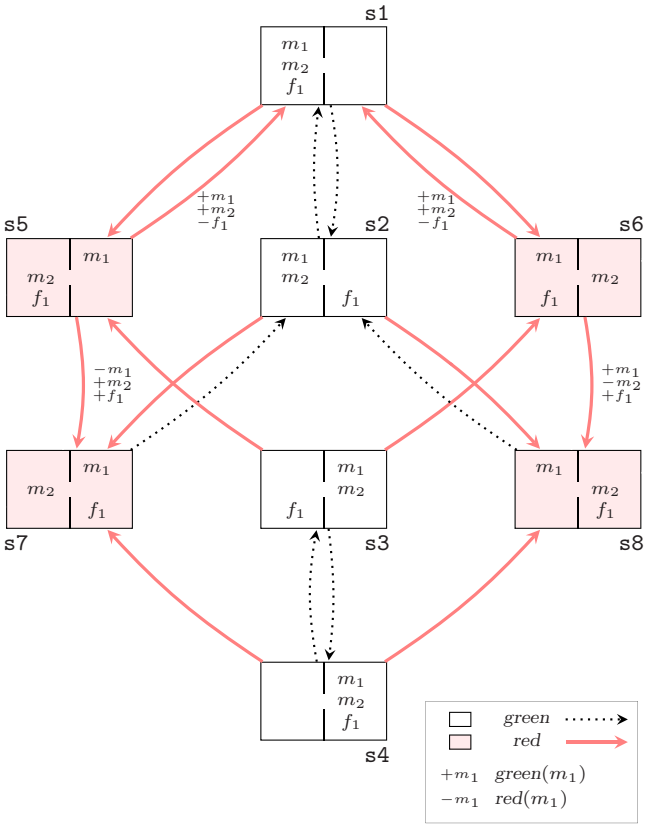


Fig. 4. System behaviour if all three agents comply with their individual norms, in as much as they can. Transitions without annotations are coloured *green*(x) for each agent x. Reflexive arcs on green nodes are omitted from the diagram.

at the upper right of the diagram in Fig. 2, in which m_1 moves to the right and m_2 and f_1 stay in the room on the right, was previously *sub-standard*(m_1). It is no longer *sub-standard*(m_1): there is now no *green*(m_1) transition from this state when m_2 and f_1 stay where they are.

Clearly, in this example, if m_1 is in the room on the left in a red system state, it has conflicting individual norms: one requiring it to move to the right, and one requiring it to stay where it is. It cannot comply with both, so neither action is *sub-standard*(m_1); both are *unavoidably-red*(m_1).

How m_1 should resolve this conflict is an interesting question but not one that we intend to consider here. It is also a question that only has relevance when m_1 is a deliberative agent which must reason about what to do. If m_1 is a purely reactive device, then its behaviour in this case could perhaps be predicted by examining its program code. Both of these possibilities are beyond the level of detail of agent and system behaviours modelled in this paper. In the simplest case we could eliminate the conflict by simply specifying that one norm takes precedence

over the other and adjusting the definition of *red*(x) and *green*(x) accordingly. Discussion of other possible mechanisms is beyond the scope of this paper.

Notice that, unlike the situation referred to earlier, where there was a conflict between agent-specific norms for two *different* agents, here we have a conflict between agent-specific norms for the *same* agent. It would be reasonable to say that there should be no conflicts of this type in any well-defined set of agent-specific norms.

There is thus a special category of *unavoidably-red*(x) transitions in which *every* action performed by x is *red*(x).

- A *red*(x) transition τ in R is *degenerately-red*(x) iff for every transition $\tau' \in R$ such that $prev(\tau) = prev(\tau')$ we have $\mathcal{M}, \tau' \models red(x)$.

Clearly

$$degenerately\text{-}red(x) \subseteq unavoidably\text{-}red(x)$$

When a transition τ is *degenerately-red*(x) then its initial state $s = prev(\tau)$ is such that there is no action that can be performed by x in compliance with its individual norms. We call such a state a *red*(x)*-sink*:

- state s is a *red*(x)*-sink* iff for every transition $\tau \in R$ such that $prev(\tau) = s$ we have $\mathcal{M}, \tau \models red(x)$.

In formulas, a state s in a model $\mathcal{M}$ is a *red*(x)-sink when:

$$\mathcal{M}, s \models \neg\langle green(x)\rangle\top \quad \text{equivalently} \quad \mathcal{M}, s \models [green(x)]\bot$$

green(x) is not 'executable' in a *red*(x)-sink. Intuitively, s is a *red*(x)-sink iff every transition τ from s is *degenerately-red*(x). A well-designed set of agent-specific norms should contain no *red*(x)-sinks. We can test for the presence of such states but we will not assume that they cannot occur. (Notice that a *red*(x)-sink is not necessarily a *red*(y)-sink for all other agents y.) There are no *red*(x)-sinks and no *degenerately-red*(x) transitions in the 'rooms' example we have been discussing. (Though there are, as we have observed, states in which there are no transitions of type $green(m_1) \wedge green(m_2) \wedge green(f_1)$.)

Similarly, we can say that a system state s is a (global) *red-sink* if there is no transition from s that is globally green. A state s is thus a red-sink when

$$\mathcal{M}, s \models \neg\langle trans{=}green\rangle\top \quad \text{equivalently} \quad \mathcal{M}, s \models [trans{=}green]\bot$$

trans=green is not 'executable' in a red-sink state.

One might think that any well designed set of system norms will have no red-sinks. That is not so. The local-global coherence constraint (if it is adopted) means that every *red*(x)-sink is also a red-sink. But even if there are no *red*(x)-sinks there can still be global red-sinks—that is one of the points we are making with the rooms example. Red-sink states may be undesirable/unwanted but we do not want to insist that they cannot occur. They can occur even in a well-designed set of agent-specific and system norms.

As an aside, note that a red-sink state can be (globally) green: a green state from which all transitions are red (or from which there are no transitions at all) is a red-sink state. We have considered extending the 'green-green-green' constraint: we could say that any transition to a (global) red-sink is undesirable/unwanted and should therefore be (globally) red. That seems natural and straightforward but its implications remain for future investigation and are not built-in to the framework as we have it now.

5 Agent-Stranded Transition Systems

Although the transition systems as they currently stand allow us to colour transitions *green*(x) and *red*(x), we are only able to give informal definitions of concepts such as *sub-standard*(x) and *unavoidably-red*(x). This is because there is no way of referring to an individual agent's actions in a transition. There is no explicit concept of an individual agent in the semantics at all. We would like to be able to extract from, or represent in, a transition system that it is specifically one agent's actions that are responsible for, or the cause of, a transition's having a certain property φ.

Let Ag be a (finite) set of agent names. An *agent-stranded LTS* is a structure

$$\langle S, A, R, Ag, strand\rangle$$

where $\langle S, A, R\rangle$ is an LTS. Models are structures $\mathcal{M} = \langle S, A, R, Ag, strand, h^{\mathrm{f}}, h^{\mathrm{a}}\rangle$ where h^{f} and h^{a} are the valuation functions for the propositional atoms of σ^{f} and σ^{a}, as before.

The new component is *strand*, which is a function on $Ag \times A$. $strand(x, \varepsilon)$ picks out from a transition label/event ε the component or 'strand' that corresponds to agent x's contribution to the event ε. We will write ε_x for $strand(x, \varepsilon)$. For example, where $Ag = \{1, \ldots, n\}$, the transition labels A may, but need not, be tuples

$$A \subseteq A_1 \times \cdots A_i \times \cdots \times A_n \times A_{\mathrm{env}}$$

where each A_i represents the possible actions of the agent i and A_{env} represents possible actions in the environment. Transition labels (events) with this structure are often used in the literature on multi-agent systems and distributed computer systems. In that case, *strand* would be defined so that

$$(a_1, \ldots, a_i, \ldots, a_n, a_{\mathrm{env}})_i = a_i$$

However, it is not necessary to restrict attention to transition labels/events of that particular form. All we require is that there is a function *strand* defined on $Ag \times A$ which picks out unambiguously an agent x's contribution to an event/transition label ε of A. As usual, ε_x may represent several concurrent actions by x, or actions with non-deterministic effects (by which we mean that there could be transitions τ and τ' with $prev(\tau) = prev(\tau')$, $\varepsilon_x = \varepsilon'_x$ where ε and ε' are the labels of τ and τ' respectively, and $post(\tau) \neq post(\tau')$).

Similarly, given a transition τ in R and an agent x in Ag, we can speak of x's strand, τ_x, of the transition τ. Agent x's strand of a transition τ is that of the transition label/event of τ:

$$\tau_x =_{\text{def}} \mathit{strand}(x, \mathit{label}(\tau))$$

τ_x may be thought of as the actions of agent x in the transition τ, where this does *not* imply that τ_x necessarily represents deliberate action, or action which has been freely chosen by x.

We do not, at this stage, introduce more granularity into the structure of states or consider norms which regulate the (local) state of an individual agent. These are possible developments for further work. Our interest here is to study the norm-governed *behaviour* of agents, and how this may be related to the norms pertaining to the system as a whole. To that end, we will concentrate on the transitions which are used to represent agents' actions.

We are now able to formalise the sub-standard and unavoidably-red categories of non-compliant behaviours, amongst other things. But first we turn to a fundamental feature of agent-specific norms that we were unable to discuss previously.

We assume as before that there is a constant *status* in σ^{f} for colouring states (globally) *red* or *green*, an event constant *trans* in σ^{a} for colouring transitions (globally) *red* or *green*, and (Boolean) event constants $\mathit{green}(x)$ and $\mathit{red}(x)$ in σ^{a} for each agent x in Ag, with $\mathit{red}(x)$ as an abbreviation for $\neg \mathit{green}(x)$.

We impose the *ggg* constraint for the global colourings representing system norms, but not for the colourings representing agent-specific norms. The local-global coherence constraint $\mathcal{M} \models \mathit{red}(x) \rightarrow \mathit{red}$ is optional. However, we do impose the following constraint on agent-specific colourings: if τ is a $\mathit{green}(x)$ (resp., $\mathit{red}(x)$) transition from a state s in model $\mathcal{M}$, then every transition τ' from state s in which agent x behaves in the same way as it does in τ must also be $\mathit{green}(x)$ (resp., $\mathit{red}(x)$). In other words, for all transitions τ and τ' in a model $\mathcal{M}$, and all agents $x \in Ag$:

$$\text{if } \mathit{prev}(\tau){=}\mathit{prev}(\tau') \text{ and } \tau_x{=}\tau'_x \text{ then } \mathcal{M}, \tau \models \mathit{green}(x) \text{ iff } \mathcal{M}, \tau' \models \mathit{green}(x) \quad (4)$$

(And hence also $\mathcal{M}, \tau \models \mathit{red}(x)$ iff $\mathcal{M}, \tau' \models \mathit{red}(x)$ whenever $\mathit{prev}(\tau) = \mathit{prev}(\tau')$ and $\tau_x = \tau'_x$.) This reflects the idea that whether actions of agent x are in accordance with the agent-specific norms for x depends only on x's actions, not on the actions of other agents, nor actions in the environment, nor other extraneous factors: we might, with appropriate philosophical caution, think of this constraint as an insistence on the absence of 'moral luck'.

Notice that the constraint (4) covers the case where $\mathit{label}(\tau) = \mathit{label}(\tau')$, that is to say, the case where there are transitions τ and τ' with $\mathit{prev}(\tau) = \mathit{prev}(\tau')$ and $\mathit{label}(\tau) = \mathit{label}(\tau')$ but different resulting states $\mathit{post}(\tau) \neq \mathit{post}(\tau')$: the event $\varepsilon = \mathit{label}(\tau)$ is non-deterministic in the state $s = \mathit{prev}(\tau)$. Constraint (4) requires that, for every agent x, both of these transitions are coloured the same way by agent-specific norms for x.

To take a simple example: suppose that when x fires a loaded gun at y, the action may result in the killing of y, or the shot may miss, or the gun may

misfire, and y survives: the shooting action is non-deterministic. We may take the view, as system designers, that a shooting transition is *red* if it results in the killing of y, and *green* if it does not. However, since x's action is the same whether the shooting is fatal or not, an agent-specific norm for x must either make both transitions $\mathit{green}(x)$ or both $\mathit{red}(x)$.

We are not putting this forward as a general principle of morality or ethics. It is a practical matter. The intention is that, in the setting of a multi-agent system of independently acting agents, the agent-specific norms for x are effective in guiding x's actions only if they are formulated in terms of what agent x can actually perceive/sense and the actions it can itself perform. At the level of detail treated here we are not modelling perceptual/sensing capabilities or actions performable by an agent explicitly. These features can be added but raise more questions than we have space for here. We leave that refinement for another occasion. For now, we insist on the 'absence of moral luck' constraint (4) as a minimal requirement for agent-specific norms.

Sub-standard behaviours

We are now able to formalise the notion of sub-standard and unavoidably-red behaviours of agent x.

Definition. Let $\mathcal{M} = \langle S, A, R, Ag, \mathit{strand}, h^{\mathrm{f}}, h^{\mathrm{a}} \rangle$ be an agent-stranded model, with event constants $\mathit{green}(x)$ and $\mathit{red}(x)$ representing the agent-specific norms for every x in Ag.

Let *unavoidably-red* and *sub-standard* be functions from the set of agents Ag to $\wp(R)$. For every agent $x \in Ag$ and every transition $\tau \in R$:

- $\tau \in \mathit{unavoidably\text{-}red}(x)$ iff $\mathcal{M}, \tau \models \mathit{red}(x)$ and, for every transition $\tau' \in R$ such that $\mathit{prev}(\tau) = \mathit{prev}(\tau')$ and $\tau_y = \tau'_y$ for every agent $y \in Ag \setminus \{x\}$, we have $\mathcal{M}, \tau' \models \mathit{red}(x)$.
- $\tau \in \mathit{sub\text{-}standard}(x)$ iff $\mathcal{M}, \tau \models \mathit{red}(x)$ and there exists $\tau' \in R$ such that $\mathit{prev}(\tau) = \mathit{prev}(\tau')$ and $\tau_x \neq \tau'_x$ and $\tau_y = \tau'_y$ for every agent $y \in Ag \setminus \{x\}$ and $\mathcal{M}, \tau' \models \mathit{green}(x)$.

Notice that the definition of $\mathit{sub\text{-}standard}(x)$ makes reference to agent x acting differently in the transitions τ and τ'. If we assume the 'absence of moral luck' property (4)—as we do—then the definition can be simplified. If $\mathcal{M}, \tau \models \mathit{red}(x)$ and $\mathcal{M}, \tau' \models \mathit{green}(x)$ for a transition τ' from the same initial state as τ ($\mathit{prev}(\tau) = \mathit{prev}(\tau')$) then the condition $\tau_x \neq \tau'_x$ is implied: if $\tau_x = \tau'_x$ then the 'absence of moral luck' constraint would be violated. The following simpler definition is equivalent to the original:

- $\tau \in \mathit{sub\text{-}standard}(x)$ iff $\mathcal{M}, \tau \models \mathit{red}(x)$ and there exists $\tau' \in R$ such that $\mathit{prev}(\tau) = \mathit{prev}(\tau')$ and $\tau_y = \tau'_y$ for every agent $y \in Ag \setminus \{x\}$ and $\mathcal{M}, \tau' \models \mathit{green}(x)$.

We will use this simpler definition of $\mathit{sub\text{-}standard}(x)$ from now on. Agent-specific colourings must satisfy the 'absence of moral luck' property; without it the notion of $\mathit{sub\text{-}standard}(x)$ is not meaningful.

One can see from the definitions that, as indicated informally earlier, every *red*(x) transition is *sub-standard*(x) if and only if it is not *unavoidably-red*(x), and that every *red*(x) transition is either *sub-standard*(x) or *unavoidably-red*(x), but not both. In other words

$$\textit{sub-standard}(x) = \textit{red}(x) \setminus \textit{unavoidably-red}(x)$$

Recall that there is a special sub-category of *degenerately-red*(x) transitions in which *every* action performed by x is *red*(x). Since *degenerately-red*(x) $\subseteq$ *unavoidably-red*(x), the *red*(x) transitions can be partitioned into three disjoint sub-types:

- *sub-standard*(x)
- *degenerately-red*(x)
- *unavoidably-red*(x) $\setminus$ *degenerately-red*(x)

We do not give a name to this third category: a well-formed set of agent-specific norms will have no *degenerately-red*(x) transitions, and then it is only the distinction between *sub-standard*(x) and *unavoidably-red*(x) that matters.

There are a number of other questions that we might now consider. For instance:

- Are there any other categories of non-compliant behaviour that could usefully be identified?
- Is it meaningful to talk about *sub-standard*(x) behaviour of an agent y other than x? What could this mean?
- If a transition is (globally) red, can we determine which of the agents, if any, is responsible for that transition's being (globally) red? In the 'rooms' example, if agent m_2 exits a room and leaves m_1 and f_1 alone together, can we determine which of the agents, if any, violated the system norms?
- If a transition is *unavoidably-red*(x) (but not *degenerately-red*(x)) is it possible to identify the subset of agents Ag whose actions prevent x from complying with its agent-specific norms?

The last question concerns forms of *collective* action/agency that will not be addressed in this paper. They are investigated elsewhere [3]. The first three questions are answered below. However the present notation is too cumbersome. We now extend the language so these and other properties can be expressed as *formulas*.

6 A Modal Language for Agency in Transitions

In this section we introduce a modal language for talking about the agent-specific components of transitions (their 'strands'). We extend the transition formulas of Sect. 2 with a (unary) operator $[_{\text{alt}}]$, and (unary) operators $[x]$ and $[\backslash x]$ for every agent $x \in Ag$. This will allow us to express concepts such as *sub-standard*(x) and *unavoidably-red*(x), and others, as formulas. In Sect. 6.2 we will introduce two 'brings it about' modalities.

6.1 A Logic of Agent Strands

Let $\mathcal{M} = \langle S, A, R, Ag, strand, h^{\mathrm{f}}, h^{\mathrm{a}} \rangle$ be an agent-stranded LTS model.

$$\mathcal{M}, \tau \models [\mathrm{alt}]\varphi \quad \text{iff} \quad \mathcal{M}, \tau' \models \varphi \text{ for every } \tau' \in R \text{ such that } prev(\tau) = prev(\tau').$$

$\langle \mathrm{alt} \rangle$ is the dual of $[\mathrm{alt}]$.

$[\mathrm{alt}]\varphi$ is satisfied by, or 'true at', a transition τ when all alternative transitions *from the same initial state* as τ satisfy φ. $\langle \mathrm{alt} \rangle \varphi$ is true at a transition τ if there exists an alternative transition from the same initial state as τ of type φ.

$[\mathrm{alt}]$ is a normal modality of type S5. In particular, we have validity (in every agent-stranded LTS) of the schemas:

$$[\mathrm{alt}]\varphi \rightarrow \varphi$$
$$[\mathrm{alt}]\varphi \rightarrow [\mathrm{alt}][\mathrm{alt}]\varphi$$
$$\neg[\mathrm{alt}]\varphi \rightarrow [\mathrm{alt}]\neg[\mathrm{alt}]\varphi$$

Clearly the following is valid

$$0{:}F \rightarrow [\mathrm{alt}]\, 0{:}F$$

Now we add the (unary) operators $[x]$ and $[\backslash x]$ for every agent $x \in Ag$.

$$\mathcal{M}, \tau \models [x]\varphi \quad \text{iff} \quad \mathcal{M}, \tau' \models \varphi \text{ for every } \tau' \in R \text{ such that } prev(\tau) = prev(\tau') \text{ and } \tau_x = \tau'_x;$$
$$\mathcal{M}, \tau \models [\backslash x]\varphi \quad \text{iff} \quad \mathcal{M}, \tau' \models \varphi \text{ for every } \tau' \in R \text{ such that } prev(\tau) = prev(\tau') \text{ and } \tau_y = \tau'_y \text{ for every } y \in Ag \setminus \{x\}.$$

$\langle x \rangle$ and $\langle \backslash x \rangle$ are the respective duals.

As in the case of $[\mathrm{alt}]$, $[x]$ and $[\backslash x]$ are used to talk about properties of alternative transitions from the same initial state: those, respectively, in which x and $Ag \setminus \{x\}$ behave in the same way. We thus have validity of:

$$[\mathrm{alt}]\varphi \rightarrow [x]\varphi \qquad [\mathrm{alt}]\varphi \rightarrow [\backslash x]\varphi$$

We will say, for short, that when $[x]\varphi$ is true at a transition τ, φ is necessary for how x acts in τ; and when $[\backslash x]\varphi$ is true at τ, that φ is necessary for how the agents $Ag \setminus \{x\}$ collectively act in τ. (Which is not the same as saying that they act together, i.e., as a kind of coalition or collective agent. We are not discussing genuine collective agency in this paper.) $\langle x \rangle \varphi$ is true at a transition τ if there is a transition τ' of type φ from the same initial state as τ in which x acts in the same way as it does in τ. Clearly $\varphi \rightarrow \langle x \rangle \varphi$ is valid. $\varphi \wedge \langle x \rangle \neg\varphi$ is true at a transition τ if τ is of type φ but there is an alternative transition of type $\neg\varphi$ from the same initial state as τ in which x acts in the same way as it does in τ. $\varphi \wedge \langle x \rangle \neg\varphi$ is equivalent to $\varphi \wedge \neg[x]\varphi$. And similarly, $\varphi \wedge \langle \backslash x \rangle \neg\varphi$ is true at

a transition τ if τ is of type φ and there is an alternative transition of type $\neg\varphi$ from the same initial state as τ in which every other agent besides x acts in the same way as it does in τ.

$[x]$ and $[\backslash x]$ are also normal modalities of type S5, so we have validity (in every agent-stranded LTS) of the schemas:

$$\begin{array}{cc} [x]\varphi \rightarrow \varphi & [\backslash x]\varphi \rightarrow \varphi \\ [x]\varphi \rightarrow [x][x]\varphi & [\backslash x]\varphi \rightarrow [\backslash x][\backslash x]\varphi \\ \neg[x]\varphi \rightarrow [x]\neg[x]\varphi & \neg[\backslash x]\varphi \rightarrow [\backslash x]\neg[\backslash x]\varphi \end{array}$$

It also follows immediately from the satisfaction definitions that the following schema is valid for all pairs of distinct agents $x \neq y$ in Ag:

$$[y]\varphi \rightarrow [\backslash x]\varphi \qquad (x \neq y)$$

equivalently, as long as Ag is not a singleton, $Ag \neq \{x\}$:

$$\bigvee_{y \in Ag \setminus \{x\}} [y]\varphi \rightarrow [\backslash x]\varphi \qquad (Ag \neq \{x\})$$

The other direction is *not* valid:

$$\not\models [\backslash x]\varphi \rightarrow \bigvee_{y \in Ag \setminus \{x\}} [y]\varphi$$

This is important. Here is a simple example. Consider the (green) state in the 'rooms' example in which all three agents are on the left (this is the state `s1` in the diagrams), and the transition τ from that state in which the female f_1 moves to the right. The resulting state is also green, and so the transition τ is (globally) green too (*trans=green* is true at τ). Clearly in all transitions from `s1` in which f_1 moves (there is only one), *trans=green* is true, and so $[f_1]$*trans=green* is true at τ. $[\backslash f_1]$*trans=green* is also true at τ. There are two transitions from `s1` in which m_1 and m_2 both act as they do in τ: τ itself, and the transition in which m_1 and m_2 stay where they are and so does f_1. Both of these transitions have *trans=green* true.

But consider $[m_1]$*trans=green*. There are three transitions from state `s1` in which m_1 acts as it does in τ: τ itself, the transition in which f_1 moves to the right and m_1 and m_2 stay where they are, and the transition in which m_2 moves to the right and m_1 and f_1 stay where they are. The last of these is a transition from a green system state to a red system state and so is of type *trans=red*. So $[m_1]$*trans=green* is false at τ. By exactly the same argument $[m_2]$*trans=green* is false at τ as well. So here we have an example where $[\backslash f_1]$*trans=green* is true but neither $[m_1]$*trans=green* nor $[m_2]$*trans=green* is true. In general $[\backslash x]\varphi$ is true at a transition τ because φ is necessary for how the agents $Ag \setminus \{x\}$ *collectively* act in τ, but that is not the same as saying that $[y]\varphi$ is true at τ for some individual agent $y \in Ag \setminus \{x\}$.

For the special case where there are exactly two agents in Ag, $Ag = \{x, y\}$, the following is valid

$$[\backslash x]\varphi \leftrightarrow [y]\varphi \qquad (Ag = \{x, y\})$$

But that is merely a special case. For the special case of a singleton set of agents $Ag = \{x\}$ we have validity of

$$[\backslash x]\varphi \leftrightarrow [_{\text{alt}}]\varphi \qquad (Ag = \{x\})$$

and hence also of $[\backslash x]\varphi \rightarrow [x]\varphi$.

The language can be generalised to allow expressions $[G]\varphi$ for any $G \subseteq Ag$. $[x]\varphi$ is then shorthand for $[\{x\}]\varphi$, $[\backslash x]\varphi$ is shorthand for $[Ag\backslash\{x\}]\varphi$, and $[_{\text{alt}}]\varphi$ is shorthand for $[\emptyset]\varphi$. The generalisation actually simplifies the technical development but since we are not discussing technical details in this paper we will not use the generalised form $[G]\varphi$ in what follows. We will note only that the logic of these operators is very familiar: the logic of $[G]\varphi$ is exactly that of 'distributed knowledge' (of type $S5$) of a group of agents G. (See e.g. [18].) Soundness, completeness, and complexity results are immediately available. We leave further discussion of technical properties to one side. See [3] for details. Our aim in this paper is to illustrate the expressiveness and uses of the language.

Examples. The 'absence of moral luck' constraint (4) for an agent x with respect to its agent-specific colouring $red(x)$ in a model $\mathcal{M}$ can be expressed as the validities:

$$\mathcal{M} \models \ red(x) \rightarrow [x]\, red(x)$$
$$\mathcal{M} \models \ green(x) \rightarrow [x]\, green(x)$$

A transition τ in a model $\mathcal{M}$ is *unavoidably-red*(x) when

$$\mathcal{M}, \tau \models [\backslash x]\, red(x)$$

It is *degenerately-red*(x) when

$$\mathcal{M}, \tau \models [_{\text{alt}}]\, red(x)$$

and hence *unavoidably-red*(x) but not *degenerately-red*(x) when

$$\mathcal{M}, \tau \models [\backslash x]\, red(x) \wedge \neg [_{\text{alt}}]\, red(x)$$

What about that category of non-compliance where an agent x could have complied with its agent-specific norms but did not, or what we called *sub-standard*(x) behaviour earlier? Expressing the definition given earlier as a formula, transition τ in a model $\mathcal{M}$ is *sub-standard*(x) when

$$\mathcal{M}, \tau \models red(x) \wedge \langle \backslash x \rangle\, green(x)$$

that is, equivalently, when:

$$\mathcal{M}, \tau \models red(x) \wedge \neg [\backslash x]\, red(x)$$

Consider now the 'absence of moral luck' constraint in a model $\mathcal{M}$, that is, the validity $\mathcal{M} \models red(x) \rightarrow [x]\, red(x)$. Agent-specific colourings must have this

property as the minimal requirement for agent-specific norms of the type we are discussing. With this constraint we have $\mathcal{M} \models \mathit{red}(x) \leftrightarrow [x]\mathit{red}(x)$, and this in turn means that a transition τ in a model $\mathcal{M}$ is *sub-standard*(x) when

$$\mathcal{M}, \tau \models [x]\mathit{red}(x) \wedge \neg[\backslash x]\mathit{red}(x)$$

Implicit in the definition of *sub-standard*(x) is the idea that it is x, rather than some other agent y, who is responsible (perhaps unintentionally or even unwittingly) for the transition's being *red*(x): it is x's actions in the transition that are the cause, unintentional or not, of the transition's being *red*(x). We now make this aspect of *sub-standard*(x) explicit. We do this by introducing two new defined operators for expressing *agency* of an agent x in bringing it about that a transition is of a particular type.

E_x and E^+_x are defined operators:

$$\begin{array}{lll} \mathrm{E}_x\varphi & =_{\mathrm{def}} & [x]\varphi \wedge \neg[_{\mathrm{alt}}]\varphi \\ \mathrm{E}^+_x\varphi & =_{\mathrm{def}} & [x]\varphi \wedge \neg[\backslash x]\varphi \end{array}$$

Both may be read as expressing a sense in which x brings it about that (a transition is of type) φ. We will explain the difference between them below. Essentially, E^+_x takes into account possible actions by other agents whereas E_x does not but treats them merely as part of the environment in which x acts.

With the 'absence of moral luck' constraint, a transition τ in a model $\mathcal{M}$ is *sub-standard*(x) when

$$\mathcal{M}, \tau \models \mathrm{E}^+_x\mathit{red}(x)$$

So, a transition is *sub-standard*(x) when x brings it about that, or is responsible for, the transition's being *red*(x).

The notation $\mathrm{E}_x\varphi$ is chosen because its definition bears a very strong resemblance to Ingmar Pörn's [2] logic of 'brings it about'—*except that* in Pörn's logic $\mathrm{E}_x p$ is used to express that agent x brings about the *state of affairs* represented by p. We are using $\mathrm{E}_x\varphi$ to express that x 'brings it about' that a *transition* has the property represented by φ. Pörn's logic does not have the analogue of $\mathrm{E}^+_x\varphi$. There are nevertheless some striking similarities, but also some very significant technical differences. See [3] for further discussion.

What about $\mathrm{E}_x\mathit{red}(x)$? What kind of non-compliant behaviour does that express? $\mathrm{E}_x\mathit{red}(x)$ is $[x]\mathit{red}(x) \wedge \neg[_{\mathrm{alt}}]\mathit{red}(x)$. Assuming the 'absence of moral luck' property for *red*(x), which we do, this is equivalent to $\mathit{red}(x) \wedge \neg[_{\mathrm{alt}}]\mathit{red}(x)$, which is just *red*$(x)$ but not *degenerately-red*(x) behaviour.

Other categories of non-compliant behaviours can similarly be expressed and investigated. To take just one example, we might look at $\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red})$ and $\mathrm{E}_x(\mathit{trans}{=}\mathit{red})$ which express that an agent x brings it about, or is responsible for, a transition's being (globally) red. These are not representations of agent-specific norms. Although $\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red})$ and $\mathrm{E}_x(\mathit{trans}{=}\mathit{red})$ both satisfy the required 'absence of moral luck' property—both of the following are valid in any model $\mathcal{M}$:

$$\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red}) \rightarrow [x]\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red})$$
$$\mathrm{E}_x(\mathit{trans}{=}\mathit{red}) \rightarrow [x]\mathrm{E}_x(\mathit{trans}{=}\mathit{red})$$

we are regarding this property as the *minimal* requirement for agent-specific norms; the other requirements, concerned with what an agent can actually sense/perceive and what actions it can actually perform, are not modelled at the level of detail we have in the present framework. The point is that $\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red})$ and $\mathrm{E}_x(\mathit{trans}{=}\mathit{red})$ are unlikely to satisfy these other requirements, since both are expressed in terms of a global transition property ($\mathit{trans}{=}\mathit{red}$) and this is not something that an individual agent is likely to be able to sense/perceive. On the other hand, $\mathrm{E}^+_x(\mathit{trans}{=}\mathit{red})$ and $\mathrm{E}_x(\mathit{trans}{=}\mathit{red})$ both express properties that might be of interest from the system designer's point of view. We will see other examples of similar properties when we look at some examples later.

Finally, as one last illustration, we might ask whether it is ever meaningful to talk about *sub-standard*(x) behaviour of an agent y other than x, that is, whether there can be transitions of type $\mathrm{E}_y\mathrm{E}^+_x \mathit{red}(x)$ or $\mathrm{E}^+_y\mathrm{E}^+_x \mathit{red}(x)$ for agents $x \neq y$. Certainly the simpler expressions $\mathrm{E}^+_y \mathit{red}(x)$ and $\mathrm{E}_y \mathit{red}(x)$ are meaningful for pairs of agents $x \neq y$ and may also represent properties of agent-specific colourings/norms that are of interest from the system designer's point of view. But *sub-standard*(x) behaviour of an agent $y \neq x$ is different: it is easy to check (as we will see later) that $\mathrm{E}_y\mathrm{E}^+_x \mathit{red}(x)$ and $\mathrm{E}^+_y\mathrm{E}^+_x \mathit{red}(x)$ are not satisfiable in any model $\mathcal{M}$; both of the following are valid

$$\neg\mathrm{E}_y\mathrm{E}^+_x \mathit{red}(x) \quad \text{and} \quad \neg\mathrm{E}^+_y\mathrm{E}^+_x \mathit{red}(x) \qquad (x \neq y)$$

No agent y can bring about, or be responsible for, a transition's being *sub-standard(x)* other than x itself.

6.2 A Logic of 'Brings It about'

For every agent $x \in Ag$, we have two defined 'brings it about' operators:

$$\mathrm{E}_x\varphi \;=_{\mathrm{def}}\; [x]\varphi \wedge \neg[\mathrm{alt}]\varphi$$
$$\mathrm{E}^+_x\varphi \;=_{\mathrm{def}}\; [x]\varphi \wedge \neg[\backslash x]\varphi$$

The study of logics of this type has a very long tradition. In computer science the best known examples are perhaps the 'stit' ('seeing to it that') family (see e.g. [19,20,21]). Segerberg [22] provides a summary of early work in this area, and Hilpinen [23] an overview of the main semantical devices that have been used, in 'stit' and other approaches. As Hilpinen observes: "The expression 'seeing to it that A' usually characterises deliberate, intentional action. 'Bringing it about that A' does not have such a connotation, and can be applied equally well to the unintentional as well as intentional (intended) consequences of one's actions, including highly improbable and accidental consequences." Our agency modalities are of this latter 'brings it about' kind. They are intended to express unintentional, perhaps even unwitting, consequences of an agent's actions, as well as possibly intentional (intended) ones.

We will not present a full account of the logical properties of the agency operators E_x and E^+_x here. They are those one would intuitively expect of 'brings it about' modalities, and are broadly in line with what is found in the literature on the logic of agency.

We will simply remark that the definitions of E_x and E^+_x have two ingredients typical of logics of agency. The first conjunct is a 'necessity condition': $\mathcal{M}, \tau \models [x]\varphi$ says that all transitions from $\mathit{prev}(\tau)$ in which x acts in the same way as it does in τ are of type φ, or as we also say, that φ is necessary for how x acts in τ. The other component is used to capture the concept of *agency* itself—the fundamental idea that φ is, in some sense, caused by or is the result of actions by x. Most accounts of agency introduce a negative 'counteraction' or counterfactual condition for this purpose, to express that had x not acted in the way that it did then the world would, or might, have been different. The second conjunct in the definition of E_x adds the 'counteraction' requirement: had x acted differently, then the transition might have been different. The conjunct $\neg[_{\mathrm{alt}}]\varphi$ says only that the transition might have been of type $\neg\varphi$: it is equivalent to $\langle_{\mathrm{alt}}\rangle\neg\varphi$. But in conjunction with the necessity condition $[x]\varphi$ it can be true at τ only if x acts differently than in τ. Thus, $\mathrm{E}_x\varphi$ is true at a transition τ if and only if φ is necessary for how x acts in τ, and had x acted differently than in τ then the transition from $\mathit{prev}(\tau)$ might have been different (i.e., of type $\neg\varphi$). For E^+_x, the counteraction condition is stronger: had x acted differently than in τ then the transition from $\mathit{prev}(\tau)$ might have been different, *even if all other agents*, besides x, had acted in the same way as they did in τ.

Both $\mathrm{E}_x\varphi$ and $\mathrm{E}^+_x\varphi$ express a sense in which agent x is 'responsible for' or 'brings it about that' (a transition is) φ. Clearly the following is valid:

$$\mathrm{E}^+_x\varphi \rightarrow \mathrm{E}_x\varphi$$

What is the difference? It is easy to check that, because $[y]\varphi \rightarrow [\backslash x]\varphi$ is valid for any $x \neq y$, the following is valid

$$\mathrm{E}^+_x\varphi \rightarrow \neg\mathrm{E}_y\varphi \qquad (x \neq y)$$

and hence also:

$$\mathrm{E}^+_x\varphi \rightarrow \neg\mathrm{E}^+_y\varphi \qquad (x \neq y)$$

So $\mathrm{E}^+_x\varphi$ expresses that it is x, and x *alone*, who brings it about that φ. In contrast, $\mathrm{E}_x\varphi$ leaves open the possibility that some other agent $y \neq x$ also brings it about that φ: the conjunction $\mathrm{E}_x\varphi \wedge \mathrm{E}_y\varphi$ can be true even when $x \neq y$.

One might feel uncomfortable with the idea that two distinct agents, acting independently, can both be responsible for 'bringing about' the same thing. But it is easy to find examples. The 'rooms' example has several instances, as will be demonstrated in Sect. 7. Notice that the conjunction $\mathrm{E}_x\varphi \wedge \mathrm{E}_y\varphi$ is equivalent to

$$[x]\varphi \wedge [y]\varphi \wedge \neg[_{\mathrm{alt}}]\varphi$$

Suppose that two agents are both pushing against a spring-loaded door and thereby keeping it shut. Suppose either one of them is strong enough by itself to keep the door shut. Both are then 'bringing it about' that the door is shut, or rather, that the transition is a 'keeping the door shut' transition. If x pushes, the door remains shut; if y pushes, the door remains shut. But 'keeping the door shut' is not unavoidable; there is a transition, viz., the one in which neither x nor y push, in which the door springs open. It is sufficient that it merely *might* spring open.

The conjunction $\mathrm{E}_x\varphi \wedge \mathrm{E}_y\varphi$ $(x \neq y)$ does *not* represent that x and y are acting in concert, or even that they are aware of each other's existence. We might as well be talking about two blind robots who have got themselves in a position where both are pushing against the same spring-loaded door. Neither can detect the other is there. This is not, and is not intended to be, a representation of genuine collective agency. The logic of (unwitting) collective action/agency is investigated in [3]. We do not have space to summarise that here.

In the same vein, there has been some discussion in the literature on whether the expression 'x brings it about that some other agent y brings it about that' is well formed. In the present framework, $\mathrm{E}_x\mathrm{E}_y\varphi$ when $x \neq y$ is well formed. We can see that it is, and examples can readily be found to demonstrate that it is meaningful. The 'keeping the door shut' example is easily modified.

As it turns out, the 'transfer of agency' property:

$$\mathrm{E}_x\mathrm{E}_y\varphi \rightarrow \mathrm{E}_x\varphi \tag{5}$$

is valid for E_x. Informally, it says that if x acts in such a way that it unwittingly brings it about that y unwittingly brings it about that φ, then x also unwittingly brings it about that φ. What of E^+_x and E^+_y for different x and y? $\mathrm{E}^+_x\mathrm{E}^+_y\varphi$ is syntactically well formed, but it is not meaningful, in the sense that the following is valid (for $x \neq y$):

$$\neg\mathrm{E}^+_x\mathrm{E}^+_y\varphi \qquad (x \neq y)$$

No agent x can by itself bring it about that some other agent y by itself brings something about. Moreover both of the following are also valid (for $x \neq y$):

$$\neg\mathrm{E}^+_x\mathrm{E}_y\varphi \qquad \neg\mathrm{E}_x\mathrm{E}^+_y\varphi \qquad (x \neq y)$$

As for 'transfer of (sole) agency', $\mathrm{E}^+_x\mathrm{E}^+_y\varphi \rightarrow \mathrm{E}^+_x\varphi$ *is* valid, but only trivially so: for any $x \neq y$, $\mathrm{E}^+_x\mathrm{E}^+_y\varphi \rightarrow \bot$ is valid, and so therefore, trivially, is $\mathrm{E}^+_x\mathrm{E}^+_y\varphi \rightarrow \mathrm{E}^+_x\varphi$.

Clearly E_x and E^+_x express a notion of *successful* action: if agent x brings it about that (a transition is of type) φ then it is indeed the case that φ. Or to put it another way (paraphrasing Hilpinen [23] quoting Chellas [24]): x can be held responsible for its being the case that φ only if it is the case that φ. E_x and E^+_x are both 'success' operators: both of the following schemes are valid:

$$\mathrm{E}_x\varphi \rightarrow \varphi \qquad \mathrm{E}^+_x\varphi \rightarrow \varphi$$

Sergot [3] examines other properties of these 'brings it about' operators and provides a sound and complete axiomatisation of the logic. Further details can be found there. They are not essential for the purposes of this paper.

6.3 Example: 'The Others Made Me Do It'

Claims that 'the others made me do it' are common in disputes about the ascription of responsibility. Merely as an illustration of the language, here are three different senses in which it can be said that 'the others made me do it'.

One possibility:

$$[x]\varphi \wedge [\backslash x]\varphi \wedge \neg[_{\text{alt}}]\varphi \tag{6}$$

This might be read as 'x did φ, but the others $Ag \setminus \{x\}$ between them acted in such a way as to make φ unavoidable'. It can be checked that (6) is equivalent to

$$\mathrm{E}_x\varphi \wedge \neg \mathrm{E}^+_x\varphi \tag{7}$$

This might be read as saying 'x did φ, but was not solely responsible'.

'The others made me do it': another possibility:

$$[\backslash x][x]\varphi \wedge \neg[_{\text{alt}}]\varphi \tag{8}$$

We mean by this that between them the others $Ag \setminus \{x\}$ acted in such a way as to make it necessary for what x does that the transition is φ. Again this does not imply any joint action, or even that the agents $Ag \setminus \{x\}$ are aware of each other's existence, or of x's. The second conjunct is because the others did not 'do' φ if there was no alternative for them, or for anyone else. In the case of a singleton set $Ag = \{x\}$ there are no 'others' and the expression (8) is false. (8) can be expressed equivalently as

$$[\backslash x]\mathrm{E}_x\varphi \tag{9}$$

Moreover, the following is valid:

$$(\mathrm{E}_x\varphi \wedge \neg \mathrm{E}^+_x\varphi) \rightarrow [\backslash x]\mathrm{E}_x\varphi$$

In other words, 'the others made me do it' (8)–(9) implies 'the others made me do it' (6)–(7), but not the other way round.

A third possibility would be to say that 'the others made me do it' means that there is some individual agent $y \in Ag \setminus \{x\}$ who brought it about that $\mathrm{E}_x\varphi$, in other words that the following is true:

$$\bigvee_{y \in Ag \setminus \{x\}} \mathrm{E}_y\mathrm{E}_x\varphi \tag{10}$$

Now, $\models \mathrm{E}_y\mathrm{E}_x\varphi \rightarrow [y]\mathrm{E}_x\varphi$ and $\models [y]\mathrm{E}_x\varphi \rightarrow [\backslash x]\mathrm{E}_x\varphi$ $(y \neq x)$. So (10) implies, but is not implied by, (9).

In summary: we can distinguish at least three different senses in which it can be said that 'the others made me do it': the third (10) implies the second (8)–(9) which implies the first (6)–(7).

6.4 Bringing about and Sustaining

$\mathrm{E}_x\varphi$ and $\mathrm{E}^+_x\varphi$ represent that x brings it about that a transition is of type φ. This is unusual. Usually, logics of agency do not talk about properties of transitions in this way. What falls in the scope of a 'brings it about' or 'sees to it that' operator is a formula representing a *state of affairs*: an agent 'brings it about' or 'sees to it that' such-and-such a state of affairs exists. How might this sense of 'brings it about' be expressed using the resources of the language presented here?

$\mathrm{E}_x(0{:}F \wedge 1{:}G)$ expresses that x brings about a transition from a state where F holds to one where G holds, and $\mathrm{E}^+_x(0{:}F \wedge 1{:}G)$ that x is solely responsible for such a transition. $\mathrm{E}_x 1{:}F$ and $\mathrm{E}^+_x 1{:}F$ express that x brings about (resp., solely) that a transition results in a state where F holds. These formulas express *one sense* in which it might be said that x 'brings about' such-and-such a state of affairs F. It is not the only sense, because it says that F holds in the state immediately following the transition, whereas we might want to say merely that F holds at some (unspecified) state in the future. Logics of agency usually do not insist that what is brought about is immediate; indeed, since transitions are not elements of the semantics, references to 'immediate' or the 'next state' are not meaningful. There is one other essential difference: $\mathrm{E}_x 1{:}F$ and $\mathrm{E}^+_x 1{:}F$ are *transition* formulas; they cannot be used to say that in a particular state s, x brings it about that such-and-such a state of affairs F holds. This sense of 'brings it about' can be expressed as a *state formula*. We omit the details. It is transitions that are of primary interest in this paper.

What about $\mathrm{E}_x 0{:}F$ and $\mathrm{E}^+_x 0{:}F$? These are not meaningful: neither is satisfiable in any model $\mathcal{M}$. Clearly, $\models 0{:}F \rightarrow [\mathrm{alt}]0{:}F$, and we have $\models [\mathrm{alt}]\varphi \rightarrow \neg\mathrm{E}_x\varphi$. However, $[\mathrm{alt}]\varphi \wedge \mathrm{E}_x\varphi' \rightarrow \mathrm{E}_x(\varphi \wedge \varphi')$ is also valid (and similarly for E^+_x), so the following pair are valid:

$$0{:}F \wedge \mathrm{E}_x 1{:}G \;\leftrightarrow\; \mathrm{E}_x(0{:}F \wedge 1{:}G)$$
$$0{:}F \wedge \mathrm{E}^+_x 1{:}G \;\leftrightarrow\; \mathrm{E}^+_x(0{:}F \wedge 1{:}G)$$

This seems very satisfactory: if in a transition where F holds in the initial state, x brings it about that G holds in the resulting state, then x brings it about that the transition is a transition from a state where F to a state where G, and vice versa.

Now, this observation makes it possible to formalise, in a rather natural way, some suggestions by Segerberg [22] and Hilpinen [23] following an idea of von Wright [25,26]. We will follow the terminology of Hilpinen's version; the others are essentially the same. Hilpinen sketches an account with two components: first, that actions are associated with transitions between states; and second, to provide the counterfactual 'counteraction' condition required to capture the notion of agency, a distinction between transitions corresponding to the agent's activity from transitions corresponding to the agent's inactivity. The latter are transitions where the agent lets 'nature take its own course'. There are then eight possible modes of agency, and because of the symmetry between F and $\neg F$, four basic forms to consider:

- x brings it about that F ($\neg F$ to F, x active);
- x lets it become the case that F ($\neg F$ to F, x inactive);

– x sustains the case that F (F to F, x active);
– x lets it remain the case that F (F to F, x inactive).

The first two correspond to a transition from a state where $\neg F$ to a state where F. The first is a type of bringing about that F by agent x; the second corresponds to inactivity by x (with respect to F)—here the agent x lets nature take its own course. The last two correspond to a transition from a state where F to a state where F. Again, the first of them is a type of bringing about that F by agent x; the second corresponds to inactivity by x (with respect to F).

As discussed by Segerberg and Hilpinen there remain a number of fundamental problems to resolve in this account. Moreover, not discussed by those authors, the picture is considerably more complicated when there are the actions of other agents to take into account and not just the effect of nature's taking its course. However, these distinctions are easily, and rather naturally, expressed in the language we have presented here.

The first ('brings it about that') and third ('sustains the case that') are straightforward: they are

$$\begin{array}{rcl} \mathrm{E}_x(0{:}\neg F \wedge 1{:}F) & \quad \text{or} \quad & \mathrm{E}^+_x(0{:}\neg F \wedge 1{:}F) \\ \mathrm{E}_x(0{:}F \wedge 1{:}F) & \quad \text{or} \quad & \mathrm{E}^+_x(0{:}F \wedge 1{:}F) \end{array}$$

respectively, depending on whether it is x's sole agency that we want to express or not.

The second and fourth cases, where x is inactive, can be expressed as:

$$\begin{array}{c} (0{:}\neg F \wedge 1{:}F) \wedge \neg \mathrm{E}_x(0{:}\neg F \wedge 1{:}F) \\ (0{:}F \wedge 1{:}F) \wedge \neg \mathrm{E}_x(0{:}F \wedge 1{:}F) \end{array}$$

(Or as above, but with E^+_x in place of E_x.)

It remains to check that these latter expressions do indeed correspond to what Hilpinen was referring to by his term 'inactive'. Whether or not that is the case, other, finer distinctions can be expressed. For example (we do not give an exhaustive exploration of all the possibilities here), supposing that $0{:}\neg F$ is true and that the transition to $1{:}F$ is not unavoidable or inevitable (in other words, that $\neg[\mathrm{alt}]1{:}F$ is true), then we can distinguish:

$$\begin{array}{l} \mathrm{E}^+_x(0{:}\neg F \wedge 1{:}F) \\ \mathrm{E}_x(0{:}\neg F \wedge 1{:}F) \wedge \neg \mathrm{E}^+_x(0{:}\neg F \wedge 1{:}F) \\ 0{:}\neg F \wedge \neg[x]1{:}F \wedge [\backslash x]1{:}F \\ 0{:}\neg F \wedge 1{:}F \wedge \neg[x]1{:}F \wedge \neg[\backslash x]1{:}F \end{array}$$

The reading of the first two is clear. The third and fourth both say that x lets it become the case that F; the first of them says that the other agents between them act in such a way that it becomes the case that F, and the last one that 'nature takes its own course'. And similarly for the 'sustains' and 'lets it remain' transitions, i.e., those of type $0{:}F \wedge 1{:}F$.

Note that intuitively x brings it about that F *simpliciter*, $\mathrm{E}_x 1{:}F$, should be equivalent to the disjunction of 'x brings it about that F' in Hilpinen's terminology and 'x sustains the case that F'. This is easily confirmed:

$$\begin{aligned} \models \mathrm{E}_x 1{:}F &\leftrightarrow (0{:}F \vee \neg 0{:}F) \wedge \mathrm{E}_x 1{:}F \\ &\leftrightarrow (0{:}F \wedge \mathrm{E}_x 1{:}F) \vee (\neg 0{:}F \wedge \mathrm{E}_x 1{:}F) \\ &\leftrightarrow \mathrm{E}_x(0{:}F \wedge 1{:}F) \vee \mathrm{E}_x(0{:}\neg F \wedge 1{:}F) \end{aligned}$$

(and likewise for E^+_x).

As an example of some of the things we might want to express using formulas of this kind consider transitions of type $0{:}\mathit{status}{=}\mathit{red} \wedge 1{:}\mathit{status}{=}\mathit{green}$. These correspond to a recovery from a red system state to a green system state. $\mathrm{E}_x(0{:}\mathit{status}{=}\mathit{red} \wedge 1{:}\mathit{status}{=}\mathit{green})$ expresses that agent x brings it about that the system recovers to a green system state, $\mathrm{E}_x(0{:}\mathit{status}{=}\mathit{red} \wedge 1{:}\mathit{status}{=}\mathit{red})$ that agent x sustains the case that the system is in a red state, $\mathrm{E}_x(0{:}\mathit{status}{=}\mathit{green} \wedge 1{:}\mathit{status}{=}\mathit{green})$ that agent x sustains the case that the system is in a green state, $\mathrm{E}_x(0{:}\mathit{status}{=}\mathit{green} \wedge 1{:}\mathit{status}{=}\mathit{red})$ that agent x brings it about, not necessarily by itself, that the system moves from a green state to a red state, and so on for the other categories where x is inactive (x lets it become the case that the system is in a red state, x lets it remain the case that the system is in a red state, and so on). We write E^+_x in place of E_x if we wish to express that x is the sole agent responsible in each case.

7 Example (Rooms, Contd)

This section illustrates how the formal language presented in the paper may be applied to the analysis of the 'rooms' example. It presents a transcript of the outputs from the iCCalc system. These transcripts are produced by specifying a list of formulas expressing properties of interest. iCCalc evaluates these formulas on all transitions in the example. It is also possible to specify formulas to be evaluated on states. We show only a small extract of state annotations here to keep the transcripts manageable.

We have also modified the example slightly. In the version discussed here, the agent-specific norms apply only to those male agents and female agents who are in a room alone together, and not, as in the previous version, to male agents and female agents in other rooms as well. So concretely: in this version, whenever a male agent x and a female agent y are alone in a room together, a transition from that state is $\mathit{green}(x)$ if the male agent x moves to the left, if there is a room to the left, $\mathit{green}(x)$ if it does not move when there is no room to the left, and $\mathit{red}(x)$ otherwise; it is $\mathit{green}(y)$ for the female agent y with 'left' replaced by 'right'. There could be several such rooms in a system state (though not in the simple example where there are just two rooms and three agents); the agent-specific norms apply to all such male-female pairs. All other transitions are $\mathit{green}(x)$ for all agents x. All other features of the example are exactly as before. The system norms colour any state where there is a male agent

and female agent alone in a room (globally) red (*status=red*); all other system states are (globally) green (*status=green*). Transitions are coloured (globally) red (*trans=red*) by the *ggg* constraint and by the local-global coherence constraint that every *red*(x) transition is also globally red; all other transitions are globally green (*trans=green*). We also have the physical constraint that no more than one agent can pass through the same doorway in any one transition. If there are many interconnecting rooms, agents could pass through different doorways in the same transition, but no more than one through any single doorway at the same time. In the simple example to be considered here, where there are just two rooms as before, this cannot happen.

There is nothing particularly significant about the change in the example. The version discussed here is arguably more realistic, since it requires only that agents are able to detect when they are alone in a room with a member of the opposite sex; there is no need to assume that klaxons or other devices exist to inform agents that the situation has arisen in other rooms. The main reason for choosing the modified version, however, is simply that features of the original example, including in particular what is *sub-standard*(x) and *unavoidably-red*(x) there, have already been discussed. The modified version provides a slightly different example.

The states and transitions for the modified version are exactly the same as those for the original. The global colouring of states is the same; the global colouring of transitions is slightly different because that is partly determined by the local-global coherence constraint and in this version of the example the agent-specific norms are different. We include a diagram of the transition system in Fig. 5 for ease of reference.

Notice that there are symmetries in the transition system because of symmetry in the example, between the two male agents m_1 and m_2, and between left and right. For that reason it is sufficient to look at transitions from just four states of the system and not all of them. We will show the transcripts for the transitions from the states in the top right quadrant of diagrams, that is, the two green states labelled `s1` and `s2` in the diagram, and the two red states `s6` and `s8`. Properties of the other states and transitions in the system are easily reconstructed by interchanging m_1 and m_2, or left and right, as the case may be.

We might begin by checking whether there are *degenerately-red*(x) transitions in the system, or (globally) red sink states. Here are the relevant queries and the output produced by ICCALC for the example:

```
?-  satisfiable [-]:red(X) where agent(X).
** not satisfiable

?-  satisfiable -executable(trans=green).
** not satisfiable
```

We trust that the ICCALC syntax is sufficiently close to the syntax of formulas used in the paper that it requires no explanation. (`[-]` is the syntax for $[_{\text{alt}}]$.) `executable`(φ) is shorthand for $\langle\varphi\rangle\top$. The first of the queries above is a transition formula asking whether there are any *degenerately-red*(x) transitions. The second is a state formula asking whether there are any (globally) red sink states.

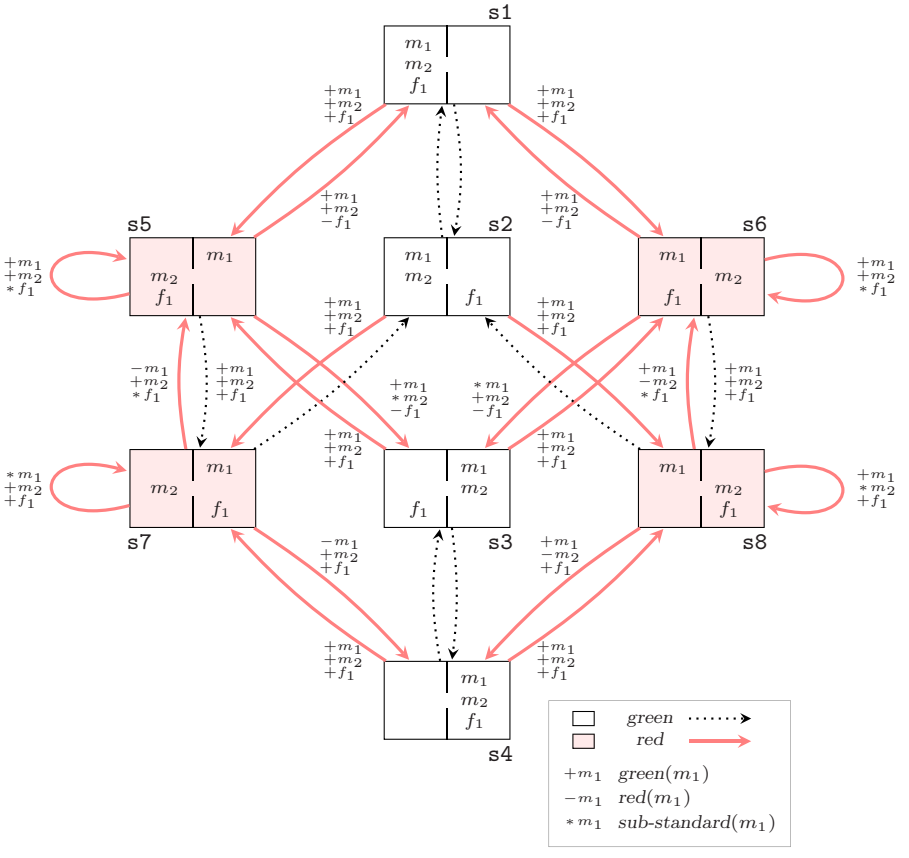


Fig. 5. The modified 'rooms' example. Transitions without annotations are coloured *green*(x) for each agent x. Reflexive arcs on green nodes are omitted from the diagram.

(The query for *red*(x)-sinks would be `satisfiable -executable(green(X))`. There are no *red*(x)-sinks in the example.)

Here we see a difference between this version of the example and the original. As discussed earlier, the original version does have (globally) red sinks. There are two of them: one where m_1 and f_1 are on the left and m_2 is on the right (state `s6`), and another (by symmetry) where m_2 and f_1 are on the left and m_1 is on the right (state `s5`). These are not global red sinks in the modified example because, unlike in the original, when m_1 and f_1 are on the left and m_2 is on the right the agent-specific norms for m_2 do not require it to move left. In the original version of the example there are no globally green (*trans=green*) transitions from these states because of the local-global coherence constraint.

Further: in the original there are states from which there is no transition unless at least one agent fails to comply with its agent-specific norms. The following ICCalc query on the original version of the example

```
?-  satisfiable -executable(green(m1) & green(m2) & green(f1)).
```

finds two states where the formula is satisfied: they are also the two global red sinks. One can check the equivalence as follows:

```
?-  valid -executable(green(m1) & green(m2) & green(f1))
                                    <-> -executable(trans=green).
** valid
```

Note that this is not the same as:

```
?-  valid (green(m1) & green(m2) & green(f1)) <-> (trans=green).
** not valid
```

In the modified version of the example, in contrast, we get

```
?-  satisfiable -executable(green(m1) & green(m2) & green(f1)).
** not satisfiable
```

Now let us look at the transcripts from ICCALC when we ask for annotations of the transitions (for the modified version of the example). We will consider first transitions from the two green states `s1` and `s2`.

```
**transition t17:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
 :[m1:move=r,green(m1),green(m2),green(f1),trans=red]
1:[loc(m1)=r,loc(m2)=l,loc(f1)=l,alone(m2,f1),status=red]
    E+(m1):(trans=red)
    E+(m1):(0:(status=green) & 1:(status=red))

**transition t18:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
 :[m2:move=r,green(m1),green(m2),green(f1),trans=red]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
    E+(m2):(trans=red)
    E+(m2):(0:(status=green) & 1:(status=red))

**transition t19:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
 :[f1:move=r,green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
    E(f1):(0:(status=green) & 1:(status=green))

**transition t20:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
 :[green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
```

Fig. 6. Transitions from the green state `s1` (all three agents on the left)

Figure 6 shows the transitions from the state `s1`, where all three agents are on the left. The numbering of the transitions in the transcript is not significant. These identifiers are generated by ICCALC when the transition system is calculated from the $nC+$ formulation of the example. They are included merely for ease of reference.

There are no *unavoidably-red*(x) or *sub-standard*(x) transitions from this state. The state is green and so the agent-specific norms in the example do not impose any constraints on how the agents may move. However, one can see that in transitions `t17` and `t18`, where one of the male agents moves to the right and leaves the other alone with the female, the one who moves is (solely) responsible for bringing it about that the transition is globally red (*trans=red*). In both cases the male who moves is also (solely) responsible for bringing it about that the system state becomes red (*status=red*) in Hilpinen's sense.

In contrast, when the female agent f_1 moves to the right (transition `t19`) she is responsible for sustaining the case that the system state is green (*status=green*). She is not solely responsible for sustaining it, however, since it also depends on how the male agents act: if the male agents both act as they do in `t19` (neither moves) then the system state remains green whether the female agent f_1 moves or not. The transition `t20` is the one where no agent moves in this state. There is nothing that we particularly want to say about it.

Now let us look at the other green state, `s2`. Figure 7 shows the transitions from this state. Although this state is not symmetrical to `s1` (the male agents and the female agent are in separate rooms here) the annotation of the transitions turns out to be the same as for `s1`. (There could of course be a difference if we specified a more extensive set of formulas to appear in annotations of transitions.)

Now let us look at the state `s6` where m_1 and f_1 are on the left and m_2 is on the right. Here the system is in a red state and the agent-specific norms impose some constraints on the behaviours of m_1 and f_1. Unlike the original version of the example, there are no agent-specific norms constraining m_2's behaviours in this state since m_2 is in a different room from the other two.

The annotation produced by ICCALC for this state is as follows:

```
**state s6: [loc(m1)=l, loc(m2)=r, loc(f1)=l, alone(m1,f1), status=red]
    oblig(m1,-m1:move) = executable(-m1:move) & -permitted(m1,-(-m1:move))
    prohib(m1,m1:move=r) = executable(m1:move=r) & -permitted(m1,m1:move=r)
    oblig(f1,f1:move=r) = executable(f1:move=r) & -permitted(f1,-f1:move=r)
    prohib(f1,-f1:move) = executable(-f1:move) & -permitted(f1,-f1:move)
```

The (Boolean) state constant *alone*(m_1, f_1) has the obvious interpretation. It is convenient to include *alone*(x, y) constants in $nC+$ formulations of larger versions of the example, where there are many rooms and more agents.

The state annotation also shows some further notational abbreviations that we find convenient. Let α be a formula of σ^{a}, that is, a propositional formula of event atoms. It is natural to say that α is permitted for x in a state s according to the agent-specific norms for x when there is a transition of type α from s which is *green*(x). Accordingly, we define:

$$\mathit{permitted}(x, \alpha) =_{\mathrm{def}} \mathit{executable}(\alpha \wedge \mathit{green}(x))$$

```
**transition t13:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
 :[m1:move=r,green(m1),green(m2),green(f1),trans=red]
1:[loc(m1)=r,loc(m2)=l,loc(f1)=r,alone(m1,f1),status=red]
    E+(m1):(trans=red)
    E+(m1):(0:(status=green) & 1:(status=red))

**transition t14:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
 :[m2:move=r,green(m1),green(m2),green(f1),trans=red]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
    E+(m2):(trans=red)
    E+(m2):(0:(status=green) & 1:(status=red))

**transition t16:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
 :[f1:move=l,green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
    E(f1):(0:(status=green) & 1:(status=green))

**transition t15:
0:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
 :[green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
```

Fig. 7. Transitions from the green state `s2` (m_1 and m_2 on the left, f_1 on the right)

Here, as usual, φ is 'executable' means only that there exists a transition of type φ from the current state: $executable(\varphi)$ is shorthand for the state formula $\langle\varphi\rangle\top$. In practice, α in an expression $permitted(x, \alpha)$ will always be a propositional formula of atoms of the form $x{:}a{=}v$.

We can define a sense of 'obligatory' and 'prohibited' action in similar fashion. As a first stab, an event of type α is *prohibited* for x in a state s according to the agent-specific norms for x if every transition of type α from state s is $red(x)$. However, that would mean that if there is no transition of type α in state s at all then α is prohibited for x. It is more informative if we add that there must be at least one transition of type α from s:

$$prohib(x, \alpha) =_{\text{def}} executable(\alpha) \wedge \neg executable(\alpha \wedge green(x))$$

(where 'executable' has its usual meaning). The above is equivalently expressed as

$$prohib(x, \alpha) =_{\text{def}} executable(\alpha) \wedge \neg permitted(x, \alpha)$$

which is the form that appears in the state annotation shown.

Similarly, it is natural to say that α is *obligatory* for x in a state s according to the agent-specific norms for x when there is at least one transition of type α from state s, and every *green*(x) transition from s is of type α (equivalently, there are no *green*(x) transitions from state s of type $\neg\alpha$). This can be expressed as:

$$\mathit{oblig}(x,\alpha) =_{\text{def}} \mathit{executable}(\alpha) \wedge \neg \mathit{executable}(\neg\alpha \wedge \mathit{green}(x))$$

which is also equivalent to:

$$\mathit{oblig}(x,\alpha) =_{\text{def}} \mathit{executable}(\alpha) \wedge \neg \mathit{permitted}(x,\neg\alpha)$$

This is the definition that is shown in the state annotation above.

The state annotation shown may give the impression that it is not necessary to have both *oblig* and *prohib*: one seems to repeat what the other says. But that is just a feature of the simplicity of this particular example. In this particular example, an agent in the left hand room can only move to the right or stay where it is: it must do one or the other. In more complicated examples, it may have many other options, and then the difference between *oblig* and *prohib* becomes marked.

It should be noted that these defined forms express only *one* sense in which α could be said to be permitted/obligatory/prohibited for x according to the agent-specific norms for x. We do not have space to discuss any other possibilities in this paper.

The transitions from state `s6` are shown in Fig. 8. In transition `t21` the male agent m_1 moves right in contravention of the agent-specific norms that require it to stay where it is in this state. The transition is *sub-standard*(m_1) because m_1 could have complied with its agent-specific norms but does not in this transition. (It is also the case that $\mathrm{E}^+_x\mathrm{E}^+_x \mathit{red}(m1)$ is true at `t21`; the ICCALC annotation would show if it were false. Compare transition `t24` below.) Transition `t21` is also *unavoidably-red*(f_1): f_1 is prevented from complying with her agent-specific norms by the actions of others in this transition. In this particular case, the transcript shows that it is m_1's actions that prevent f_1 from moving right as required by her agent-specific norms. Transition `t21` also provides an example where two different agents (m_1 and f_1 here) both bring about, are both responsible for, the transition's being globally red (*trans=red*) and thus in contravention of the system norms. We also see that m_1 is solely responsible in this transition for bringing it about that the system state becomes green, that is, moves from a red state to a green state. So here we have an example where the system recovers from a red system state to a green system state, but where the transition itself is (globally) red, and therefore in contravention of the system norms, and where the agent m_1 who is solely responsible for the recovery from a red system state to a green system state does so by acting in contravention of its own agent-specific norms.

Transition `t22`, in which m_2 moves left, is similar but not symmetric with `t21`. Here, the agent-specific norms for m_2 do not require it to stay where it is because m_2 is not in the same room as m_1 and f_1. m_2 is thus free to move according to its agent-specific norms, but if it does move, then it makes it impossible for the female agent f_1 to comply with hers: the transition is *unavoidably-red*(f_1), and as the transcript shows, it is m_2 who is responsible (though not solely responsible)

```
**transition t21:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
 :[m1:move=r,red(m1),green(m2),red(f1),trans=red]
1:[loc(m1)=r,loc(m2)=r,loc(f1)=l,status=green]
    substandard(m1) = E+(m1):red(m1)
    unavoidably_red(f1) = [-f1]:red(f1)
        E(m1):red(f1)
    E(m1):(trans=red)
    E(f1):(trans=red)
    E+(m1):(0:(status=red) & 1:(status=green))

**transition t22:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
 :[m2:move=l,green(m1),green(m2),red(f1),trans=red]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=l,status=green]
    unavoidably_red(f1) = [-f1]:red(f1)
        E(m2):red(f1)
    E(m2):(trans=red)
    E(f1):(trans=red)
    E+(m2):(0:(status=red) & 1:(status=green))

**transition t23:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
 :[f1:move=r,green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
    E(f1):(0:(status=red) & 1:(status=red))

**transition t24:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
 :[green(m1),green(m2),red(f1),trans=red]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
    substandard(f1) = E+(f1):red(f1)
        -E+(f1):E+(f1):red(f1)
    E+(f1):(trans=red)
        -E+(f1):E+(f1):(trans=red)
```

Fig. 8. Transitions from the red state `s6` (m_1 and f_1 on the left, m_2 on the right)

for making it so. Transition `t22` is also another example of two different agents (m_2 and f_1) both bringing it about that a transition is of a particular type (globally red). m_2 is also solely responsible for bringing it about that the system recovers (becomes green), though unlike in `t21`, not in contravention of its own agent-specific norms.

Transition `t23` is straightforward. Here all three agents comply with their agent-specific norms. f_1 however, although acting in compliance with her agent-specific norms by moving to the right, nevertheless is thereby responsible (though

```
**transition t9:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
 :[m1:move=r,green(m1),red(m2),green(f1),trans=red]
1:[loc(m1)=r,loc(m2)=r,loc(f1)=r,status=green]
    unavoidably_red(m2) = [-m2]:red(m2)
        E(m1):red(m2)
    E(m1):(trans=red)
    E(m2):(trans=red)
    E+(m1):(0:(status=red) & 1:(status=green))

**transition t10:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
 :[m2:move=l,green(m1),green(m2),green(f1),trans=green]
1:[loc(m1)=l,loc(m2)=l,loc(f1)=r,status=green]
    E+(m2):(0:(status=red) & 1:(status=green))

**transition t12:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
 :[f1:move=l,green(m1),red(m2),red(f1),trans=red]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=l,alone(m1,f1),status=red]
    unavoidably_red(m2) = [-m2]:red(m2)
        E(f1):red(m2)
    substandard(f1) = E+(f1):red(f1)
    E(m2):(trans=red)
    E(f1):(trans=red)
    E(f1):(0:(status=red) & 1:(status=red))

**transition t11:
0:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
 :[green(m1),red(m2),green(f1),trans=red]
1:[loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
    substandard(m2) = E+(m2):red(m2)
        -E+(m2):E+(m2):red(m2)
    E+(m2):(trans=red)
        -E+(m2):E+(m2):(trans=red)
```

Fig. 9. Transitions from the red state `s8` (m_1 on the left, m_2 and f_1 on the right)

not solely responsible) for sustaining the case that the system remains in a red system state. As with other similar examples, one should be very careful not say that an agent behaves badly if it is responsible for sustaining, or bringing about, that a system state remains, or becomes, a red system state. It may also act well in the same transition, in the sense that it complies with its agent-specific norms. System norms and agent-specific norms are related, for instance by local-global coherence, but they express different standards of legality, acceptability,

desirability, and therefore different standards of what it means to say that an agent acts well or acts badly.

Finally, transition `t24`, in which no agent moves, is *sub-standard*(f_1) because here f_1 could have complied with her agent-specific norms but did not. She is also solely responsible for bringing about that the transition is globally red. Note though, that although $\mathrm{E}^+_{f_1}$ *red*(f_1) is true at `t24` (this is what *sub-standard*(f_1) means), $\mathrm{E}^+_{f_1}\mathrm{E}^+_{f_1}$ *red*(f_1) is not true. In general $\mathrm{E}^+_x\varphi \rightarrow \mathrm{E}^+_x\mathrm{E}^+_x\varphi$ is not valid. Here we have an example. We can see that $[f_1]\mathrm{E}^+_{f_1}$ *red*(f_1) is not true at `t24`. If it were, that would mean $\mathrm{E}^+_{f_1}$ *red*(f_1) is true at every transition from state `s6` in which f_1 acts as she does in `t24`, i.e., does not move. Transitions `t21` and `t22` are both like this, but $\mathrm{E}^+_{f_1}$ *red*(f_1) is not true at either of them: neither of them is *sub-standard*(f_1). And if $[f_1]\mathrm{E}^+_{f_1}$ *red*(f_1) is not true at `t24` then neither is $\mathrm{E}^+_{f_1}\mathrm{E}^+_{f_1}$ *red*(f_1). Similarly for $\mathrm{E}^+_{f_1}$(*trans*=*red*); $[f_1]\mathrm{E}^+_{f_1}$(*trans*=*red*) is not true at `t24`, as is easily confirmed.

To complete the picture, here is the ICCALC output for the other red state, `s8`.

```
**state s8: [loc(m1)=l,loc(m2)=r,loc(f1)=r,alone(m2,f1),status=red]
    oblig(m2,m2:move=l) = executable(m2:move=l) & -permitted(m2,-m2:move=l)
    prohib(m2,-m2:move) = executable(-m2:move) & -permitted(m2,-m2:move)
    oblig(f1,-f1:move) = executable(-f1:move) & -permitted(f1,-(-f1:move))
    prohib(f1,f1:move=l) = executable(f1:move=l) & -permitted(f1,f1:move=l)
```

The transitions from this state are shown in Fig. 9. We do not provide a commentary. Although the details are different, the general points we wish to make have already been discussed. (When a formula $\mathrm{E}^+_x\varphi$ is true, $\mathrm{E}^+_x\mathrm{E}^+_x\varphi$ is also true unless shown otherwise.)

8 Conclusion

We have presented a modal-logical language for talking about properties of states and transitions of a labelled transition system and, by introducing agent 'strands' as a component of transitions, for talking about what transition properties are necessary for how a particular agent, or group of agents, acts in a particular transition. This allows us in turn to introduce two defined 'brings it about' modalities. The novel feature is that we switch attention from talking about an agent's bringing it about that a certain state of affairs exists to talking about an agent's bringing it about that a transition has a certain property. We are thereby able to make explicit the notions of agency that underpin various forms of norm compliant or non-compliant behaviour, and to be able to discuss relationships between system norms and agent-specific norms using the formal language. The aim, amongst other things, is to be able to investigate what kind of system properties emerge if we assume, for instance, that all agents of a certain class will do the best that they can to comply with their individual norms, or never act in such a way that they make non-compliance unavoidable for others. We are also able to express when an agent, or group of agents, is responsible, solely or otherwise, for bringing about that a transition complies with system norms,

for bringing it about that the system recovers from a red system state to a green system state, for sustaining the case that the system remains in a green system state, and so on.

Besides the generalisation to (unwitting) collective agency [3] there are three main directions of current work.

(1) *Scaleability.* It might be felt that the 'rooms' example used in this paper is too simple to be taken seriously as representative of real-world domains. We deliberately chose the simplest configuration of rooms and agents that allowed us to make the points we wanted to make while still being able to be depicted in their entirety. The example works just as well with more rooms, more than two categories of agents, and a wider repertoire of actions that the agents are able to perform. Generally, the issues we have addressed arise whenever we put together a complex system of interacting agents, acting independently, whose behaviours are subject to their own agent-specific norms, and where we wish to impose further system norms to regulate possible interactions.

Nevertheless, it is clear that serious issues of scaleability remain, and that in particular we confront the same state explosion problems that arise in all modelling approaches of this kind. These are problems, however, that are the subject of extensive current research. There is nothing that prevents us from applying emerging techniques and solutions to agent-stranded transition systems too.

One promising direction that we are exploring is the use of agent-centric projections. Roughly, given a model $\mathcal{M}$ describing system behaviour, it is possible to define a projection $\mathcal{M}_x$ in which all states and transitions indistinguishable for an agent x are collapsed into one, and all other states and transitions are discarded. $\mathcal{M}_x$ thus models system behaviour from an individual agent x's perspective. Some information is lost, but (depending of course on what is indistinguishable for an individual x), the projection $\mathcal{M}_x$ is much smaller and more manageable than the full model $\mathcal{M}$.

(2) *Agent-specific norms.* One fundamental feature of agent-specific norms, as we see it, is that to be effective or even meaningful in guiding the actions of an individual agent x they must be formulated in terms of what the agent x can actually sense/perceive of its environment, and the actions that an agent x can actually perform. We referred to the 'absence of moral luck' constraint as the minimal requirement we must impose on agent-specific norms. To do a proper job it is necessary to refine and extend the semantical structures in order to model these features explicitly. This part is not so difficult. We will present the details in another paper. There is also the further question of how agent-specific norms once formulated can be incorporated into an agent's implementation—in the case of a 'lightweight' reactive agent, how to modify its program code to take agent-specific norms into account, and in the case of a deliberative agent, how to represent the agent-specific norms in a form that the agent can use in its reasoning processes. We have very little to say about that yet.

(3) *The representation of norms.* We gave in the paper one simple formulation of what it can mean to say that an action is obligatory or permitted for x

according to the agent-specific norms for x. There are many other variations and distinctions that can be expressed using the resources of the language. Generally, the logic of norms and the logic of action/agency have often been studied together, and it remains to explore how the full resources of the language can be used to articulate distinctions and issues that have previously been discussed in the literature. Further, it is well known in the field of deontic logic that a simple binary classification of states and/or transitions into green/red (ideal/sub-ideal, permitted/not permitted) is too simple to deal adequately with many kinds of norms. In [6], for instance, we presented a refinement of the current approach in which the states of a transition systems were ordered depending on how well each complied with a set of explicitly stated norms. Much more remains to be done along these lines.

Acknowledgements

The characterisation of non-compliant behaviours, agent-specific norms, the 'rooms' example, and the iccalc implementation is joint work with Robert Craven. I am grateful to Alex Artikis for helpful comments on an earlier draft.

References

1. Craven, R., Sergot, M.: Agent strands in the action language nC+. Journal of Applied Logic 6(2), 172–191 (2008)
2. Pörn, I.: Action Theory and Social Science: Some Formal Models. In: Synthese Library, Number 120. D. Reidel, Dordrecht (1977)
3. Sergot, M.: The logic of unwitting collective agency. Technical Report 2008/6, Department of Computing, Imperial College London (2008)
4. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence 153(1–2), 49–104 (2004)
5. Sergot, M.: (C+)++: An action language for modelling norms and institutions. Technical Report 2004/8, Department of Computing, Imperial College London (2004)
6. Sergot, M., Craven, R.: The deontic component of action language nC+. In: Goble, L., Meyer, J.J.C. (eds.) DEON 2006. LNCS (LNAI), vol. 4048, pp. 222–237. Springer, Heidelberg (2006)
7. Große, G., Khalil, H.: State Event Logic. Journal of the IGPL 4(1), 47–74 (1996)
8. Venema, Y.: Points, lines and diamonds: a two-sorted modal logic for projective planes. Journal of Logic and Computation 9(5), 601–621 (1999)
9. Sauro, L., Gerbrandy, J., van der Hoek, W., Wooldridge, M.: Reasoning about action and cooperation. In: Proceedings of the Fifth International Joint Conference on Autonomous agents and Multiagent Systems: AAMAS 2006, pp. 185–192. ACM, New York (2006)
10. von Wright, G.H.: Norm and Action—A Logical Enquiry. Routledge and Kegan Paul, London (1963)
11. Chellas, B.F.: Modal Logic—An Introduction. Cambridge University Press, Cambridge (1980)

12. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
13. Carmo, J., Jones, A.J.I.: Deontic database constraints, violation and recovery. Studia Logica 57(1), 139–165 (1996)
14. Meyden, R.: The dynamic logic of permission. Journal of Logic and Computation 6(3), 465–479 (1996)
15. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic 29(1), 109–136 (1988)
16. Lomuscio, A., Sergot, M.J.: Deontic interpreted systems. Studia Logica 75(1), 63–92 (2003)
17. Ågotnes, T., van der Hoek, W., Rodriguez-Aguilar, J.A., Sierra, C., Wooldridge, M.: On the logic of normative systems. In: Veloso, M.M. (ed.) Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1175–1180. AAAI Press, Menlo Park (2007)
18. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
19. Belnap, N., Perloff, M.: Seeing to it that: a canonical form for agentives. Theoria 54, 175–199 (1988)
20. Horty, J.F., Belnap, N.: The deliberative stit: a study of action, omission, ability, and obligation. Journal of Philosophical Logic 24(6), 583–644 (1995)
21. Horty, J.F.: Agency and Deontic Logic. Oxford University Press, Oxford (2001)
22. Segerberg, K.: Getting started: Beginnings in the logic of action. Studia Logica 51(3–4), 347–378 (1992)
23. Hilpinen, R.: On action and agency. In: Ejerhed, E., Lindström, S. (eds.) Logic, Action and Cognition—Essays in Philosophical Logic. Trends in Logic, Studia Logica Library, vol. 2, pp. 3–27. Kluwer Academic Publishers, Dordrecht (1997)
24. Chellas, B.F.: The Logical Form of Imperatives. Dissertation, Stanford University (1969)
25. von Wright, G.H.: An essay in deontic logic and the general theory of action. Number 21 in Acta Philosophica Fennica (1968)
26. von Wright, G.H.: Practical Reason. Blackwell, Oxford (1983)

Managing Conflict Resolution in Norm-Regulated Environments*

Martin J. Kollingbaum[1], Wamberto W. Vasconcelos[1], Andres García-Camino[2], and Tim J. Norman[1]

[1] Dept. of Computing Science, Univ. of Aberdeen, Aberdeen AB24 3UE, UK
{mkolling,wvasconc,tnorman}@csd.abdn.ac.uk
[2] IIIA-CSIC, Campus UAB 08193 Bellaterra, Spain
andres@iiia.csic.es

Abstract. Norms are the obligations, permissions and prohibitions associated with members of a society. Norms provide a useful abstraction with which to specify and regulate the behaviour of self-interested software agents in open, heterogeneous systems. Any realistic account of norms must address their dynamic nature: the norms associated with agents will change as agents act (and interact) – prohibitions can be lifted, obligations can be fulfilled, and permissions can be revoked as a result of agents' actions. These norms may at times *conflict* with one another, that is, an action may be simultaneously prohibited and obliged (or prohibited and permitted). Such conflicts cause norm-compliant agents to experience a paralysis: whatever they do (or not do) will go against a norm. In this paper we present mechanisms to detect and resolve normative conflicts. We achieve more expressiveness, precision and realism in our norms by using *constraints* over first-order variables. The mechanisms to detect and resolve norm conflicts take into account such constraints and are based on first-order unification and constraint satisfaction. We also explain how the mechanisms can be deployed in the management of norms regulating environments for software agents.

1 Introduction

Norms are the obligations, permissions and prohibitions associated with members of a society [3,18]. Norms provide a useful abstraction to specify and regulate the observable behaviour in electronic environments of self-interested, heterogeneous software agents [2,6]. Norms also support the establishment of organisational structures for coordinated resource sharing and problem solving [8,19]. Norm-regulated environments may experience problems when norms associated with their agents are in *conflict* – actions that are forbidden, may, at the same time, also be obliged and/or permitted.

* This research is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence (http://www.usukita.org).

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 55–71, 2008.

We illustrate such situations with a scenario in which software agents share information. A norm stipulates that "everyone is forbidden to share any information with agent ag_1" (that is, everyone is forbidden to $share(Info, ag_1)$). However, as agents interact, a new norm stipulates that "everyone is obliged to share a particular piece of information $info_1$ with all other agents" (that is, everyone is obliged to $share(info_1, X)$)[1]. These two norms are in conflict regarding action $share/2$ and some of its possible values. Normative conflicts "paralyse" norm-compliant software agents because whatever they do (or refrain from doing) goes against a norm.

In this paper, we propose a means to automatically detect and resolve norm conflicts. We make use of first-order unification [7] to find out if and how norms overlap in their *scope of influence* [15]. If such a conflict is detected, a resolution can be found by proposing a *curtailment* of the conflicting norms. We curtail norms by adding constraints, thus limiting their scope of influence. For example, if we add the constraint $Info \neq info_1$ to the prohibition above, we curtail this norm excluding $info_1$ from its scope of influence – the norm becomes "everyone is forbidden to share any information, excluding $info_1$, with ag_1". The scope of influence of the prohibition becomes restricted and does not overlap with the influence of the obligation. Alternatively, if we add the constraint $X \neq ag_1$ to the obligation above, we curtail its scope of influence to exclude a value, thus avoiding the conflict with the prohibition.

In next Section we present our approach to norm-governed agency. In Section 3 we define norm conflicts and how to resolve them. Section 4 presents algorithms for the management of norm-regulated environments, that is, the adoption and removal of norms. In Section 5 we explain a simple mechanism endowing agents with norm-awareness. Section 6 explores indirect conflicts arising from relationships among actions. In Section 7 we survey related work. In Section 8 we draw conclusions and give directions for future work.

2 Norm-Governed Agency

Our model of norm-governed agency assumes that agents take on roles within a society or organisation and that these roles have norms associated with them. Roles, as used in, *e.g.*, [20], help us abstract from individual agents, defining a pattern of behaviour to which any agent that adopts a role ought to conform. We shall make use of two finite, non-empty sets, $Agents = \{a_1, \ldots, a_n\}$ and $Roles = \{r_1, \ldots, r_m\}$, representing, respectively, the sets of agent identifiers and role labels. Central to our model is the concept of actions performed by agents:

Definition 1. *$\langle a : r, \bar{\varphi}, t\rangle$ represents a specific action $\bar{\varphi}$ (a ground first-order atomic formula), performed by $a \in Agents$ adopting $r \in Roles$ at time $t \in \mathbb{N}$.*

Although agents are regarded as performing their actions in a distributed fashion (thus contributing to the overall enactment of the system), we propose a global

[1] $Info, X$ are variables and $ag_1, info_1$ are constants identifying a particular agent and a particular piece of information, respectively.

account for all actions performed. It is important to record the authorship of actions and the time when they occur. The set Ξ stores such tuples recording actions of agents and represents a *trace* or a history of the enactment of a society of agents from a global point of view:

Definition 2. *A global enactment state Ξ is a finite, possibly empty, set of tuples $\langle a{:}r, \bar{\varphi}, t\rangle$.*

A global enactment state Ξ can be "sliced" into many partial states $\Xi_a = \{\langle a{:}\, r, \bar{\varphi}, t\rangle \in \Xi \,|\, a \in \mathit{Agents}\}$ containing all actions of a specific agent a. Similarly, we could have partial states $\Xi_r = \{\langle a : r, \bar{\varphi}, t\rangle \in \Xi \,|\, r \in \mathit{Roles}\}$, representing the global state Ξ "sliced" across the various roles. We make use of a global enactment state to simplify our exposition; however, a fully distributed (and thus more scalable) account of enactment states can be achieved by slicing them as above and managing them in a distributed fashion.

2.1 Norm Specification

We extend the notion of a norm as presented in [26]. We adopt the notation of [20] for specifying norms, complementing it with *constraints* [14]. Constraints are used to *refine* the influence of norms on specific actions. A syntax for constraints is introduced as follows:

Definition 3. *Constraints, represented as γ, are any construct of the form $\tau \lhd \tau'$, where τ, τ' are first-order terms (that is, a variable, a constant or a function applied to terms) and $\lhd \in \{=, \neq, >, \geq, <, \leq\}$.*

We shall make use of numbers and arithmetic functions to build terms τ. Arithmetic functions may appear infix, following their usual conventions[2]. Some sample constraints are $X < 120$ and $X < (Y + Z)$. Norms are thus defined:

Definition 4. *A norm ω is a tuple $\langle \nu, t_d, t_a, t_e\rangle$, where ν is any construct of the form $\mathsf{O}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ (an obligation), $\mathsf{P}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ (a permission) or $\mathsf{F}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ (a prohibition), where τ_1, τ_2 are terms, φ is a first-order atomic formula and γ_i, $0 \leq i \leq n$, are constraints. The components $t_d, t_a, t_e \in \mathbb{N}$ are, respectively, the time when ν was declared (introduced), when ν becomes active and when ν expires, $t_d \leq t_a \leq t_e$.*

Term τ_1 identifies the agent(s) to which the norm is applicable and τ_2 is the role of such agent(s). $\mathsf{O}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ thus represents an obligation on agent τ_1 taking up role τ_2 to bring about φ, subject to constraints γ_i, $0 \leq i \leq n$. The γ_i's express constraints on those variables occurring in φ.

In the definition above we only cater for conjunctions of constraints. If disjunctions are required then a norm must be established for each disjunct. For instance, if we required the norm $\mathsf{P}_{A:R}\mathit{move}(A) \wedge A < 10 \vee A = 15$ then we must break it into two norms $\mathsf{P}_{A:R}\mathit{move}(A) \wedge A < 10$ and $\mathsf{P}_{A:R}\mathit{move}(A) \wedge A = 15$.

[2] We adopt Prolog's convention [1] using strings starting with a capital letter to represent variables and strings starting with a small letter to represent constants.

We assume an implicit universal quantification over variables in ν. For instance, $\mathsf{P}_{A:R}p(X,b,c)$ stands for $\forall A \in Agents.\forall R \in Roles.\forall X.\mathsf{P}_{A:R}p(X,b,c)$.

We propose to formally represent the normative positions of all agents taking part in a virtual society, from a global perspective. By "normative position" we mean the "social burden" associated with individuals [12], that is, their obligations, permissions and prohibitions:

Definition 5. *A global normative state Ω is a finite and possibly empty set of tuples $\omega = \langle \nu, t_d, t_a, t_e \rangle$.*

A global normative state, expressed by Ω, complements the enactment state of a virtual society, expressed by Ξ, with information on the normative positions of individual agents. The management (*i.e.*, creation and updating) of global normative states is an interesting area of research. A practical approach is that of [11]: rules depict how norms should be inserted and removed as a result of agents' actions. A sample rule is

$$\langle Ag_1 : seller, sold(Ag_2, Good, Price), T\rangle \rightsquigarrow \oplus\langle \mathsf{O}_{Ag_2:buyer} pay(Ag_1, Price), (T+1), (T+1), (T+5)\rangle$$

representing that if an agent Ag_1 acting as a seller agrees to selling to Ag_2 some *Good* at cost *Price* then we introduce (denoted by the "$\oplus$" operator) an obligation on Ag_2 acting as a buyer, to pay Ag_1 the agreed *Price* within 5 "ticks" of a global clock. Similarly to Ξ, we use a single normative state Ω to simplify our exposition; however, we can also slice Ω into various sub-sets and manage them in a distributed fashion as explored in [9].

3 Norm Conflicts

We provide definitions for norm conflicts, enabling their detection and resolution. Constraints confer more expressiveness and precision on norms, but the mechanisms for detection and resolution must factor them in. We use first-order unification [7] and constraint satisfaction [14] as the building blocks of our mechanisms. Unification allows us *i)* to detect whether norms are in conflict and *ii)* to detect the set of actions that are under the influence of a norm. Initially, we define substitutions:

Definition 6. *A substitution σ is a finite and possibly empty set of pairs x/τ, where x is a variable and τ is a term.*

We define the application of a substitution in accordance with [7] . In addition, we describe, how substitutions are applied to norms (X stands for O, P or F):

1. $c \cdot \sigma = c$ for a constant c.
2. $x \cdot \sigma = \tau \cdot \sigma$ if $x/\tau \in \sigma$; otherwise $x \cdot \sigma = x$.
3. $p^n(\tau_0, \ldots, \tau_n) \cdot \sigma = p^n(\tau_0 \cdot \sigma, \ldots, \tau_n \cdot \sigma)$.
4. $(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i) \cdot \sigma = (\mathsf{X}_{(\tau_1\cdot\sigma):(\tau_2\cdot\sigma)}\varphi \cdot \sigma) \wedge \bigwedge_{i=0}^{n}(\gamma_i \cdot \sigma)$.
5. $\langle \nu, t_d, t_a, t_e\rangle \cdot \sigma = \langle (\nu \cdot \sigma), t_d, t_a, t_e\rangle$

A substitution σ is a *unifier* of two terms τ_1, τ_2, if $\tau_1 \cdot \sigma = \tau.\sigma$. Unification is a fundamental problem in automated theorem proving and many algorithms have been proposed [7]; recent work offers means to obtain unifiers efficiently. We shall use unification in the following way:

Definition 7. *unify*(τ_1, τ_2, σ) *holds iff* $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$, *for some* σ. *unify*$(p^n(\tau_0, \ldots, \tau_n), p^n(\tau'_0, \ldots, \tau'_n), \sigma)$ *holds iff unify*$(\tau_i, \tau'_i, \sigma), 0 \leq i \leq n$.

The *unify* relationship checks if a substitution σ is indeed a unifier for τ_1, τ_2, but it can also be used to find σ. We assume that *unify* is a suitable implementation of a unification algorithm which *i)* always terminates (possibly failing, if a unifier cannot be found); *ii)* is correct; and *iii)* has a linear computational complexity.

3.1 Conflict Detection

Conflict detection consists of checking if the variables of a prohibition and those of a permission/obligation have overlapping values. The values of the arguments of a norm specify its scope of influence, that is, which agent/role the norm concerns, and which values of the action it addresses. In Fig. 1 we show two norms over action *deploy*(S, X, Y), establishing that sensor S is to be deployed on grid position (X, Y). The norms are $\mathsf{O}_{A_1:R_1}\, deploy(s_1, X_1, Y_1) \wedge 10 \leq X_1 \leq 50 \wedge 5 \leq Y_1 \leq 45$ and $\mathsf{F}_{A_2:R_2}\, deploy(s_1, X_2, Y_2) \wedge 5 \leq X_2 \leq 60 \wedge 15 \leq Y_2 \leq 40$, their scopes shown as rectangles filled with different patterns. The overlap of their scopes is the rectangle in which both patterns appear together. Norm conflict is formally defined as follows:

Definition 8. *Norms* $\omega, \omega' \in \Omega$ *are in conflict under substitution* σ, *denoted as* **conflict**$(\omega, \omega', \sigma)$, *iff the following conditions hold:*

1. $\omega = \langle(\mathsf{F}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i), t_d, t_a, t_e\rangle$, $\omega' = \langle(\mathsf{O}'_{\tau'_1:\tau'_2}\varphi' \wedge \bigwedge_{i=0}^{n} \gamma'_i), t'_d, t'_a, t'_e\rangle$,
2. *unify*$(\langle\tau_1, \tau_2, \varphi\rangle, \langle\tau'_1, \tau'_2, \varphi'\rangle, \sigma)$, *satisfy*$(\bigwedge_{i=0}^{n} \gamma_i \wedge (\bigwedge_{i=0}^{m} \gamma'_i \cdot \sigma))$
3. *overlap*(t_a, t_e, t'_a, t'_e).

That is, a conflict occurs if *i)* a substitution σ can be found that unifies the variables of two norms[3], and *ii)* the conjunction $\bigwedge_{i=0}^{n} \gamma_i \wedge (\bigwedge_{i=0}^{m} \gamma'_i) \cdot \sigma)$ of constraints

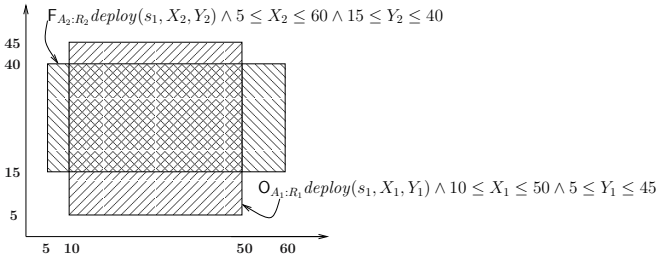


Fig. 1. Conflict Detection: Overlap in Scopes of Influence

[3] A similar definition is required to address the case of conflict between a prohibition and a permission – the first condition should be changed to $\omega' = \langle(\mathsf{P}'_{\tau'_1:\tau'_2}\varphi' \wedge \bigwedge_{i=0}^{n} \gamma'_i), t'_d, t'_a, t'_e\rangle$. The rest of the definition remains the same.

from both norms can be satisfied[4] (taking σ under consideration), and *iii)* the activation period of the norms overlap. The *overlap* relationship holds if *i)* $t_a \leq t'_a \leq t_e$; or *ii)* $t'_a \leq t_a \leq t'_e$.

For instance, $\mathsf{P}_{A:R}p(c, X) \wedge X > 50$ and $\mathsf{F}_{a:b}p(Y, Z) \wedge Z < 100$ are in conflict. We can obtain a substitution $\sigma = \{A/a, R/b, Y/c, X/Z\}$ which shows how they overlap. Being able to construct such a unifier is a first indication that there may be a conflict or *overlap* of influence between both norms regarding the defined action. The constraints on the norms may restrict the overlap and, therefore, leave actions under certain variable bindings free of conflict. We, therefore, have to investigate the constraints of both norms in order to see if an overlap of the values indeed occurs. In our example, the permission has a constraint $X > 50$ and the prohibition has $Z < 100$. By using the substitution X/Z, we see that $50 < X < 100$ and $50 < Z < 100$ represent ranges of values for variables X and Z where a conflict will occur.

For convenience (and without any loss of generality) we assume that our norms are in a special format: any non-variable term τ occurring in ω is replaced by a fresh variable X (not occurring anywhere in ω) and a constraint $X = \tau$ is added to ω. This transformation can be easily automated by scanning ω from left to right, collecting all non-variable terms $\{\tau_1, \ldots, \tau_n\}$; then we add $\wedge_{i=1}^{n} X_i = \tau_i$ to ν. For example, norm $\mathsf{P}_{A:R}p(c, X) \wedge X > 50$ is transformed into $\mathsf{P}_{A:R}p(C, X) \wedge X > 50 \wedge C = c$.

3.2 Conflict Resolution

We propose to resolve norm conflicts by manipulating the constraints on their variables, thus removing any overlap in their values. In Fig. 2 we show the norms of Fig. 1 without the intersection between their scopes of influence – the prohibition has been *curtailed*, its scope being reduced to avoid the values that the obligation addresses. Specific constraints are added to the prohibition in order to perform this curtailment; these additional constraints are derived from the obligation, as we explain below. In our example above, we obtain

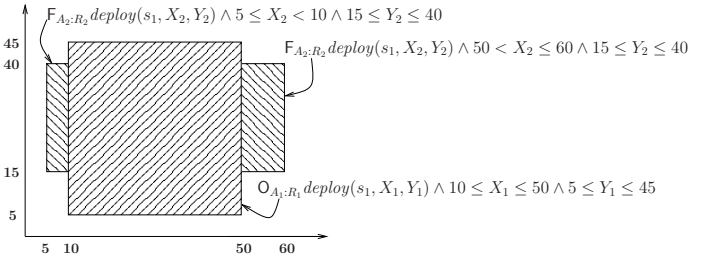


Fig. 2. Conflict Resolution: Curtailment of Scopes of Influence

[4] We assume an implementation of the *satisfy* relationship based on "off-the-shelf" constraint satisfaction libraries such as those provided by SICStus Prolog [25] and it holds if the conjunction of constraints is satisfiable.

two prohibitions, $\mathsf{F}_{A_2:R_2}\, deploy(s_1, X_2, Y_2) \wedge 5 \leq X_2 < 10 \wedge 15 \leq Y_2 \leq 40$ and $\mathsf{F}_{A_2:R_2}\, deploy(s_1, X_2, Y_2) \wedge 50 < X_2 \leq 60 \wedge 15 \leq Y_2 \leq 40$.

We formally define below how the curtailment of norms takes place. It is important to notice that the curtailment of a norm creates a new (possibly empty) set of curtailed norms:

Definition 9. *Relationship* **curtail**$(\omega, \omega', \Omega)$*, where* $\omega = \langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i, t_d, t_a, t_e\rangle$ *and* $\omega' = \langle \mathsf{X}'_{\tau_1':\tau_2'}\varphi' \wedge \bigwedge_{j=0}^{m}\gamma'_j, t'_d, t'_a, t'_e\rangle$ *(*X *and* X' *being either* O, F *or* P*) holds iff* Ω *is a possibly empty and finite set of norms obtained by curtailing* ω *with respect to* ω'*. The following cases arise:*

1. *If* **conflict**$(\omega, \omega', \sigma)$ *does not hold then* $\Omega = \{\omega\}$*, that is, the curtailment of a non-conflicting norm* ω *is* ω *itself.*
2. *If* **conflict**$(\omega, \omega', \sigma)$ *holds, then* $\Omega = \{\omega_0^c, \ldots, \omega_m^c\}$*, where* $\omega_j^c = \langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge (\neg\gamma'_j \cdot \sigma), t_d, t_a, t_e\rangle$*,* $0 \leq j \leq m$*.*

In order to curtail ω, thus avoiding any overlapping of values its variables may have with those variables of ω', we must "merge" the negated constraints of ω' with those of ω. Additionally, in order to ensure the appropriate correspondence of variables between ω and ω' is captured, we must apply the substitution σ obtained via **conflict**$(\omega, \omega', \sigma)$ on the merged negated constraints.

By combining the constraints of $\nu = \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ and $\nu' = \mathsf{X}'_{\tau_1':\tau_2'}\varphi' \wedge \bigwedge_{j=0}^{m}\gamma'_j$, we obtain the curtailed norm $\nu^c = \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\bigwedge_{j=0}^{m}\gamma'_j \cdot \sigma)$. The following equivalences hold:

$$\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\bigwedge_{j=0}^{m}\gamma'_j \cdot \sigma) \equiv \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge (\bigvee_{j=0}^{m}\neg\gamma'_j \cdot \sigma)$$

That is, $\bigvee_{j=0}^{m}(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\gamma'_j \cdot \sigma))$. This shows that each constraint on ν' leads to a possible solution for the resolution of a conflict and a possible curtailment of ν. The curtailment thus produces a set of curtailed norms $\nu_j^c = \mathsf{X}_{\tau_1:\tau_2}p(t_1, \ldots, t_n) \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg\gamma'_j \cdot \sigma, 0 \leq j \leq m$.

Although each of the $\nu_j^c, 0 \leq j \leq m$ represents a solution to the norm conflict, we advocate that *all* of them have to be added to Ω in order to replace the curtailed norm. This would allow a preservation of as much of the original scope of the curtailed norm as possible.

As an illustrative example, let us suppose $\Omega = \{\langle \mathsf{F}_{A:R}p(C, X) \wedge C = c \wedge X > 50, t_d, t_a, t_e\rangle\}$. If we try to introduce a new norm $\omega' = \langle \mathsf{P}_{B:S}p(Y, Z) \wedge B = a \wedge S = r \wedge Z > 100, t'_d, t'_a, t'_e\rangle$ to Ω, then we detect a conflict. This conflict can be resolved by curtailing one of the two conflicting norms. The constraints in ω' are used to create such a curtailment. The new permission ω' contains the following constraints: $B = a$, $S = r$ and $Z > 100$. Using σ, we construct copies of ω, but adding $\neg\gamma'_i \cdot \sigma$ to them. In our example the constraint $Z > 100$ becomes $\neg(Z > 100) \cdot \sigma$, that is, $X \leq 100$. With the three constraints contained

in ω', three options for curtailing ω can be constructed. A new Ω' is constructed, containing *all* the options for curtailment:

$$\Omega' = \left\{ \begin{array}{l} \langle \mathsf{P}_{B:S} p(Y,Z) \wedge B = a \wedge S = r \wedge Z > 100, t'_d, t'_a, t'_e \rangle \\ \langle \mathsf{F}_{A:R} p(C,X) \wedge C = c \wedge X > 50 \ \wedge A \neq a, t_d, t_a, t_e \rangle \\ \langle \mathsf{F}_{A:R} p(C,X) \wedge C = c \wedge X > 50 \ \wedge R \neq r, t_d, t_a, t_e \rangle \\ \langle \mathsf{F}_{A:R} p(C,X) \wedge C = c \wedge X > 50 \ \wedge X \leq 100, t_d, t_a, t_e \rangle \end{array} \right\}$$

For each $\neg\gamma'_i \cdot \sigma$ ($A \neq a$, $R \neq r$ and $X \leq 100$ in our example), the original prohibition is extended with one of these constraints and added as a new, more restricted prohibition to Ω'. Each of these options represents a part of the scope of influence regarding actions of the original prohibition ω, restricted in such a way that a conflict with the permission is avoided. In order to allow a check whether any other action that was prohibited by ω is prohibited or not, it is necessary to make all three prohibitions available in Ω'. If there are other conflicts, additional curtailments may be necessary.

3.3 An Implementation of Norm Curtailment

We show in Figure 3 a prototypical implementation of the curtailment process as a logic program. We show our logic program with numbered lines to enable the easy referencing of its constructs. Lines 1–7 define **curtail**, and lines 8–14 define an auxiliary predicate *merge*/3. Lines 1–6 depict the case when the norms are in conflict: the test in line 4 ensures this. Line 5 invokes the auxiliary predicate *merge*/3 which, as the name suggests, merges the conjunction of γ_i's with the negated constraints γ'_j's. Line 6 assembles Ω by collecting the members Γ of the list $\widehat{\Gamma}$ and using them to create curtailed versions of ω. The elements of the list $\widehat{\Gamma}$ assembled via *merge*/3 are of the form $(\bigwedge_{i=0}^{n} \gamma_i) \wedge (\neg\gamma'_j \cdot \sigma)$ – additionally, in our implementation we check if each element is satisfiable[5] (line 10). The rationale for this is that there is no point in creating a norm which will never be applicable as its constraints cannot be satisfied, so these are discarded during their preparation.

```
1  curtail(ω, ω′, Ω) ←
2     ω = ⟨X_{τ1:τ2} φ ∧ ⋀_{i=0}^{n} γ_i, t_d, t_a, t_e⟩∧
3     ω′ = ⟨X′_{τ′1:τ′2} φ′ ∧ ⋀_{j=0}^{m} γ′_j, t′_d, t′_a, t′_e⟩∧
4     conflict(ω, ω′, σ)∧
5     merge([(¬γ′_0 · σ), ..., (¬γ′_m · σ)], (⋀_{i=0}^{n} γ_i), Γ̂)∧
6     setof(⟨X_{τ1:τ2} φ ∧ Γ, t_d, t_a, t_e⟩, member(Γ, Γ̂), Ω)
7  curtail(ω, ω′, {ω})

8  merge([], _, [])
9  merge([(¬γ′ · σ)|Gs], (⋀_{i=0}^{n} γ_i), [Γ|Γ̂]) ←
10    satisfy((⋀_{i=0}^{n} γ_i) ∧ (¬γ′ · σ))∧
11    Γ = (⋀_{i=0}^{n} γ_i) ∧ (¬γ′ · σ)∧
12    merge(Gs, (⋀_{i=0}^{n} γ_i), Γ̂)
13 merge([_|Gs], (⋀_{i=0}^{n} γ_i), Γ̂) ←
14    merge(Gs, (⋀_{i=0}^{n} γ_i), Γ̂)
```

Fig. 3. Implementation of **curtail** as a Logic Program

[5] We have made use of SICStus Prolog [25] constraint satisfaction libraries [13].

3.4 Curtailment Policies

Rather than assuming that a specific deontic modality is always curtailed[6], we propose to explicitly use *policies* determining, given a pair of norms, which one is to be curtailed. Such policies confer more flexibility on our curtailment mechanism, allowing for a fine-grained control over how norms should be handled:

Definition 10. *A policy π is a tuple $\langle \omega, \omega', (\bigwedge_{i=0}^{n} \gamma_i) \rangle$ establishing that ω should be curtailed (and ω' should be preserved), if $(\bigwedge_{i=0}^{n} \gamma_i)$ hold.*

A sample policy is $\langle \langle \mathsf{F}_{A:R} p(X,Y), T_d, T_a, T_e \rangle, \langle \mathsf{P}_{A:R} p(X,Y), T'_d, T'_a, T'_e \rangle, (T_d < T'_d) \rangle$. It expresses that any prohibition held by any agent that corresponds to the pattern $\mathsf{F}_{A:R} p(X,Y)$ has to be curtailed, if the additional constraint, which expresses that the prohibition's time of declaration T_d precedes that of the permission's T'_d, holds. Adding constraints to policies allows us a fine-grained control of conflict resolution, capturing classic forms of deontic conflict resolution – the constraint in the example establishes a precedence relationship between the two norms known as *lex posterior* (see Section 7 for more details). We shall represent a set of such policies as Π.

4 Management of Normative States

In this section we explain how our approach to conflict detection and resolution can be used to manage normative states Ω. We explain how we preserve conflict-freedom when adopting a new norm as well as how norms are removed – when a norm is removed we must guarantee that any curtailment it caused is undone.

4.1 Norm Adoption

The algorithm in Fig. 4 describes how an originally conflict-free (possibly empty) set Ω can be extended in a fashion that resolves any emerging conflicts during norm adoption. With that, a conflict-free Ω is always transformed into a conflict-free Ω' that may contain curtailments. The algorithm makes use of a set Π of policies determining how the curtailment of conflicting norms should be done.

When a norm is curtailed, a set of new norms replace the original norm. This set of norms is collected into Ω'' by **curtail**$(\omega, \omega', \Omega'')$. A curtailment takes place if there is a conflict between ω and ω'. The conflict test creates a unifier σ re-used in the policy test. When checking for a policy that is applicable, the algorithm uses unification to check *(a)* whether ω matches/unifies with ω_π and ω' with ω'_π; and *(b)* whether the policy constraints hold under the given σ. If a previously agreed policy in Π determines that the newly adopted norm ω is to be curtailed in case of a conflict with an existing $\omega' \in \Omega$, then the new set Ω' is created by adding Ω'' (the curtailed norms) to Ω. If the policy determines a curtailment of an existing $\omega' \in \Omega$ when a conflict arises with the new norm ω, then a new set Ω' is formed by *a)* removing ω' from Ω and *b)* adding ω and the set Ω'' to Ω.

[6] In [26], for instance, prohibitions are always curtailed. This ensures the choices on the agents' behaviour are kept as open as possible.

```
algorithm adoptNorm(ω, Ω, Π, Ω')
input ω, Ω, Π
output Ω'
begin
  Ω' := ∅
  if Ω = ∅ then
    Ω' := Ω ∪ {ω}
  else
    for each ω' ∈ Ω do
      if conflict(ω, ω', σ) then // test for conflict
        if ⟨ω_π, ω'_π, (⋀_{i=0}^{n} γ_i)⟩ ∈ Π and // test policy
           unify(ω, ω_π, σ) and unify(ω', ω'_π, σ) and satisfy(⋀_{i=0}^{n}(γ_i · σ)) then
        begin
          curtail(ω, ω', Ω'')
          Ω' := Ω ∪ Ω''
        end
        else
          if ⟨ω'_π, ω_π, (⋀_{i=0}^{n} γ_i)⟩ ∈ Π and // test policy
             unify(ω, ω_π, σ) and unify(ω', ω'_π, σ) and satisfy(⋀_{i=0}^{n}(γ_i · σ)) then
          begin
            curtail(ω', ω, Ω'')
            Ω' := (Ω − {ω'}) ∪ ({ω} ∪ Ω'')
          end
end
```

Fig. 4. Norm Adoption Algorithm

4.2 Norm Removal

As well as adding norms to normative states we also need to support their removal. Since the introduction of a norm may have interfered with other norms, resulting in their curtailment, when that norm is removed we must *undo* the curtailments it caused, that is, we must return (or "roll back") to a previous form of the normative state. In order to allow curtailments of norms to be undone, we record the complete *history* of normative states representing the evolution of normative positions of agents:

Definition 11. *$\mathcal{H}$ is a non-empty and finite sequence of tuples $\langle i, \Omega, \omega, \pi\rangle$, where $i \in \mathbb{N}$ represents the order of the tuples, Ω is a normative state, ω is a norm and π is a policy.*

We shall denote the empty history as $\langle\ \rangle$. We define the concatenation of sequences as follows: if $\mathcal{H}$ is a sequence and h is a tuple, then $\mathcal{H} \bullet h$ is a new sequence consisting of $\mathcal{H}$ followed by h. Any non-empty sequence $\mathcal{H}$ can be decomposed as $\mathcal{H} = \mathcal{H}' \bullet h \bullet \mathcal{H}''$, $\mathcal{H}'$ and/or $\mathcal{H}''$ possibly empty. The following properties hold for our histories $\mathcal{H}$:

1. $\mathcal{H} = \langle 0, \emptyset, \omega, \pi\rangle \bullet \mathcal{H}'$
2. $\mathcal{H} = \mathcal{H}' \bullet \langle i, \Omega', \omega', \pi'\rangle \bullet \langle i+1, \Omega'', \omega'', \pi''\rangle \bullet \mathcal{H}''$
3. $adoptNorm(\omega_i, \Omega_i, \{\pi_i\}, \Omega_{i+1})$

The first condition establishes the first element of a history to be an empty Ω. The second condition establishes that the tuples are completely ordered on their first component. The third condition establishes the relationship between any two consecutive tuples in histories: normative state Ω_{i+1} is obtained by adding ω_i to Ω_i adopting policy π_i.

```
algorithm removeNorm(ω, H, Ω, H')
input ω, H
output Ω, H'
begin
  if H = H' • ⟨k, Ω_k, ω, π_k⟩ • ··· • ⟨n, Ω_n, ω_n, π_n⟩ then
  begin
    Ω := Ω_k
    for i = k + 1 to n do
    begin
      adoptNorm(ω_i, Ω, {π_i}, Ω')
      H' := H' • ⟨i, Ω, ω_i, π_i⟩
      Ω := Ω'
    end
  end
  else
  begin
    H = H'' • ⟨n, Ω_n, ω_n, π_n⟩
    Ω := Ω_n, H' := H
  end
end
```

Fig. 5. Algorithm to Remove Norms

$\mathcal{H}$ is required to allow the retraction of a norm in an ordered fashion, as not only the norm itself has to be removed but also all the curtailments it caused when it was introduced in Ω. $\mathcal{H}$ contains a tuple $\langle i, \Omega, \omega, \pi \rangle$ that indicates the introduction of norm ω and, therefore, provides us with a normative state Ω *before* the introduction of ω. The effect of the introduction of ω can be reversed by using Ω and redoing (performing a kind of "roll forward") all the inclusions of norms according to the sequence represented in $\mathcal{H}$ via *adoptNorm*.

This mechanism is detailed in Figure 5: algorithm *removeNorm* describes how to remove a norm ω given a history $\mathcal{H}$; it outputs a normative state Ω and an updated history $\mathcal{H}'$ and works as follows. Initially, the algorithm checks if ω indeed appears in $\mathcal{H}$ – it does so by matching $\mathcal{H}$ against a pattern of a sequence in which ω appears as part of a tuple (notice that the pattern initialises the new history $\mathcal{H}'$). If there is such a tuple in $\mathcal{H}$, then we initialise Ω as Ω_k, that is, the normative state *before* ω was introduced. Following that, the **for** loop implements a *roll forward*, whereby new normative states (and associated history $\mathcal{H}'$) are computed by introducing the $\omega_i, k+1 \leq i \leq n$, which come after ω in the original history $\mathcal{H}$. If ω does not occur in any of the tuples of $\mathcal{H}$ (this case is catered by the **else** of the **if** construct) then the algorithm uses pattern-matching to decompose the input history $\mathcal{H}$ and obtain its last tuple – this is necessary as this tuple contains the most recent normative state Ω_n which is assigned to Ω; the new history $\mathcal{H}'$ is the same as $\mathcal{H}$.

5 Norm-Aware Agent Societies

With a set Ω that reflects a conflict-free global normative situation, agents can test whether their actions are norm-compliant. In order to check actions for norm-compliance, we again use unification. If an action unifies with a norm, then it is within its scope of influence:

$$\begin{array}{l}\texttt{check}(Action, \omega) \leftarrow \\ \quad Action = \langle a : r, \bar{\varphi}, t\rangle \wedge \\ \quad \omega = \langle (\mathsf{F}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i), t_d, t_a, t_e\rangle \wedge \\ \quad unify(\langle a, r, \bar{\varphi}\rangle, \langle \tau_1, \tau_2, \varphi\rangle, \sigma) \wedge satisfy(\bigwedge_{i=0}^{n} \gamma_i \cdot \sigma) \wedge t_a \leq t \leq t_e\end{array}$$

Fig. 6. Check if Action is within Influence of a Prohibition

Definition 12. *$\langle a : r, \bar{\varphi}, t\rangle$, is within the scope of influence of $\langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i, t_d, t_a, t_e\rangle$ (where X is either O, P or F) iff the following conditions hold:*

1. *$unify(\langle a, r, \bar{\varphi}\rangle, \langle \tau_1, \tau_2, \varphi\rangle, \sigma)$ and $satisfy(\bigwedge_{i=0}^{n} \gamma_i \cdot \sigma)$*
2. *$t_a \leq t \leq t_e$*

This definition can be used to establish a predicate `check`/2, which holds if its first argument, a candidate action (in the format of the elements of Ξ of Def. 2), is within the influence of a prohibition ω, its second parameter. Figure 6 shows the definition of this relationship as a logic program. Similarly to the check of conflicts between norms, it tests *i)* if the agent performing the action and its role unify with the appropriate terms τ_1, τ_2 of ω; *ii)* if the actions $\bar{\varphi}, \varphi$ themselves unify; and *iii)* the conjunction of the constraints of both norms can be satisfied, all under the same unifier σ. Lastly, it checks if the time of the action is within the norm temporal influence.

6 Indirect Conflicts

In our previous discussion, norm conflicts were detected via a direct comparison of atomic formulae representing actions. However, conflicts and inconsistencies may also arise *indirectly* via relationships among actions. For instance, if an agent has associated norms $\mathsf{P}_{A:R}p(X)$ and $\mathsf{F}_{A:R}q(X, X)$ and that the action $p(X)$ amounts to the action $q(X, X)$, then we can rewrite the permission as $\mathsf{P}_{A:R}q(X, X)$ and identify an *indirect* conflict between the two norms. We use a set of *domain axioms* in order to declare such domain-specific relationships between actions:

Definition 13. *The set of domain axioms, denoted as Δ, are a finite and possibly empty set of formulae $\varphi \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_n)$ where $\varphi, \varphi'_i, 1 \leq i \leq n$, are atomic first-order formulae.*

In order to address indirect conflicts between norms based on domain-specific relationships of actions, we have to adapt our curtailment mechanism. With the introduction of domain axioms $\varphi \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_n)$, the conflict check has to be performed for each of the conjuncts in this relationship. For example, if we have $\Delta = \{(p(X) \rightarrow q(X, X) \wedge r(X, Y))\}$ and $\langle \mathsf{P}_{A:R}p(X), t_d, t_a, t_e\rangle$, then actions $q(X, X)$ and $r(X, Y)$ are also permitted. If we also have $\langle \mathsf{F}_{A:R}q(X, X), t_d, t_a, t_e\rangle$ then an indirect conflict occurs. We now revisit Def. 8, extending it to address indirect conflicts:

Definition 14. *An indirect conflict arises between two norms ω, ω' under a set of domain axioms Δ and a substitution σ, denoted as* $\mathbf{conflict}^*(\Delta, \omega, \omega')$*, iff:*

1. $\mathbf{conflict}(\omega, \omega', \sigma)$*, or*
2. $\omega = \langle(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle$*, there is an axiom* $(\varphi' \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_m)) \in \Delta$ *such that unify*$(\varphi, \varphi', \sigma')$*, and* $\bigvee_{i=1}^{m} \mathbf{conflict}^*(\Delta, \langle(\mathsf{X}_{\tau_1:\tau_2}\varphi'_i \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle \cdot \sigma', \omega')$,

The above definition recursively follows a chain of indirect conflicts, looking for any two conflicting norms. Case 1 provides the base case of the recursion, checking if norms ω, ω' are in direct conflict. Case 2 addresses the general recursive case: if a norm X (that is, O, P or F) on an action φ unifies with φ' on the left-hand side of a domain axiom $(\varphi \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_m)) \in \Delta$, then we "transfer" the norm from φ to $\varphi'_1, \ldots, \varphi'_m$, thus obtaining $\langle(\mathsf{X}_{\tau_1:\tau_2}\varphi'_i \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle, 1 \leq i \leq m$. If we (recursively) find an indirect conflict between ω' and at least one of these norms, then an indirect conflict arises between the original norms ω, ω'. It is important to notice that the substitution σ' that unifies φ and φ' is factored in the mechanism: we apply it to the new φ'_is in the recursive call(s).

Domain axioms may also accommodate the delegation of actions between agents. Such a delegation transfers norms across the agent community and, with that, conflicts also. We introduce a special logical operator $\varphi^{\tau_1:\tau_2\ \tau'_1:\tau'_2}(\varphi'_1 \wedge \cdots \wedge \varphi'_n)$ to represent that agent τ_1 adopting role τ_2 can transfer any norms on action φ to agent τ'_1 adopting role τ'_2, which should carry out actions $\varphi'_1 \wedge \cdots \wedge \varphi'_n$ instead. We formally capture the meaning of this operator as follows:

3. $\omega = \langle(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle$*, there is a delegation axiom* $(\varphi^{\tau_1:\tau_2\ \tau'_1:\tau'_2}(\varphi'_1 \wedge \cdots \wedge \varphi'_m)) \in \Delta$*, s.t. unify*$(\langle\varphi, \tau_1, \tau_2\rangle, \langle\varphi', \tau'_1, \tau'_2\rangle, \sigma')$*, and* $\bigvee_{i=1}^{m} \mathbf{conflict}^*(\Delta, \langle(\mathsf{X}_{\tau'_1:\tau'_2}\varphi'_i \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle \cdot \sigma', \omega')$

That is, we obtain a domain axiom and check if its action, role and agent unify with those of ω. The norm will be transferred to the new actions $(\varphi'_1 \wedge \cdots \wedge \varphi'_m)$ but these will be associated with a possibly different agent/role pair $\tau'_1{:}\tau'_2$. The new norms are recursively checked and if at least one of them conflicts with ω', then an indirect conflict arises. Means to detect loops in delegation must be added to the definition above.

7 Related Work

Efforts to keep law systems conflict-free can be traced back to the jurisprudential practice in human society. Inconsistency in law is an important issue and legal theorists use a diverse set of terms such as, for example, *normative inconsistencies/conflicts*, *antinomies*, *discordance*, etc., in order to describe this phenomenon. There are three classic strategies for resolving deontic conflicts by establishing a precedence relationship between norms: *legis posterioris* – the most recent norm takes precedence, *legis superioris* – the norm imposed by the strongest power takes precedence, and *legis specialis* – the most specific norm takes precedence [17].

Early investigations into norm conflicts were outlined in [21], describing three forms of conflict/inconsistency as *total-total*, *total-partial* and *intersection*. These are special cases of the intersection of norms as described in [16] – a permission entailing the prohibition, a prohibition entailing the permission or an overlap of both norms.

In [22,23], aspects of legal reasoning such as non-monotonic reasoning in law, negation and conflict are discussed. It is pointed out that legal reasoning is often based on *prima facie* incompatible premises, which is due to the defeasibility of legal norms and the dynamics of normative systems, where new norms may contradict older ones (principle of *legis posterioris*), the concurrence of multiple legal sources with normative power distributed among different bodies issuing contradicting norms (principle of *legis superioris*), and semantic indeterminacy. To resolve such conflicts, it is proposed to establish an ordering among norms according to criteria such as hierarchy (*legis superioris*), chronology (*legis posterioris*), speciality (exception to the norm are preferred) or hermeneutics (more plausible interpretations are preferred). The work presented in [16] discusses in part these kinds of strategies, proposing conflict resolution according to the criteria mentioned above.

The work described in [5] analyses different normative conflicts – in spite of its title, the analysis is an informal one. That work differentiates between actions that are simultaneously prohibited and permitted – these are called *deontic inconsistencies* – and actions that are simultaneously prohibited and obliged – these are called *deontic conflicts*. The former is merely an "inconsistency" because a permission may not be acted upon, so no real conflict actually occurs. On the other hand, those situations when an action is simultaneously obliged and prohibited represent conflicts, as both obligations and prohibitions influence behaviours in an incompatible fashion. Our approach to detecting deontic conflict can capture the three forms of conflict/inconsistency of [21], *viz.* total-total, total-partial and intersection, respectively, when the permission entails the prohibition, when the prohibition entails the permission and when they simply overlap. Finally, we notice that the *world knowledge* explained in [5], required to relate actions, can be formally captured by our indirect norm conflicts depicted in Section 6.

The work presented in this paper is an adaptation and extension of [16,26] and [10], also providing an investigation into deontic modalities for representing normative concepts [4,24]. In [26], a conflict detection and resolution based on unification is introduced: we build on that research, introducing constraints to the mechanisms proposed in that work.

8 Conclusions, Discussion and Future Work

We have presented mechanisms to detect and resolve conflicts in norm-regulated environment. Such conflicts arise when an action is simultaneously obliged and prohibited/permitted. We represent norms as first-order atomic formulae whose variables can have arbitrary constraints associated – this allows for more

expressive norms, with a finer granularity and greater precision. The mechanisms are based on first-order unification and constraint satisfaction, extending the work of [26], and addressing a more expressive class of norms. Our conflict resolution mechanism amounts to manipulating the constraints of norms to avoid overlapping values of variables – this is called the "curtailment" of variables/norms. A prototypical implementation of the curtailment process is given as a logic program and is used in the management of the normative state of an agent society. We have also introduced an algorithm to manage the adoption of possibly conflicting norms, whereby explicit policies depict how the curtailment between specific norms should take place, as well as an algorithm depicting how norms should be removed, thus undoing the effects of past curtailments.

In this work we only considered norms with universal quantifiers over discrete domains. These assumptions limit the applicability of our solution. Universally quantified permissions capture a common sense of norm: an agent is permitted to perform an action with any value its quantified variables may get (and which satisfy the constraints); the same holds for prohibitions. However, obligations are conventionally existential: an agent is obliged to perform an action once with one of its possible values.

We are currently exploiting our approach in mission-critical scenarios [27], including, for instance, combat and disaster recovery (*e.g.* extreme weather conditions and urban terrorism). Our goal is to describe mission scripts as sets of norms: these will work as contracts that teams of human and software agents can peruse and make sense of. Mission-critical contracts should allow for the delegation of actions and norms, via pre-established relationships between roles: we have been experimenting with special "count as" operators which neatly capture this. Additionally, our mission-critical contracts should allow the representation of plan scripts with the breakdown of composite actions into the simplest atomic actions. Norms associated with composite actions will be distributed across the composite actions, possibly being delegated to different agents and/or roles.

References

1. Apt, K.R.: From Logic Programming to Prolog. Prentice-Hall, Englewood Cliffs (1997)
2. Artikis, A., Kamara, L., Pitt, J., Sergot, M.: A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. In: Leite, J.A., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS (LNAI), vol. 3476. Springer, Heidelberg (2005)
3. Conte, R., Castelfranchi, C.: Understanding the Functions of Norms in Social Groups through Simulation. In: Gilbert, N., Conte, R. (eds.) Artificial Societies: The Computer Simulation of Social Life, pp. 252–267. UCL Press, London (1995)
4. Dignum, F.: Autonomous Agents with Norms. A.I. & Law 7, 69–79 (1999)
5. Elhag, A., Breuker, J., Brouwer, P.: On the Formal Analysis of Normative Conflicts. Information & Comms. Techn. Law 9(3), 207–217 (2000)
6. Esteva, M., Padget, J., Sierra, C.: Formalizing a Language for Institutions and Norms. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333. Springer, Heidelberg (2002)

7. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer, New York (1990)
8. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int' J. Supercomputer Applications 15(3), 209–235 (2001)
9. Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.-A., Vasconcelos, W.: Distributed Norm Management in Regulated Multi-agent Systems. In: Procs. 6th Int'l Joint Conf. on Autonomous Agents & Multiagent Systems (AAMAS 2007), Honolulu, Hawai'i (May 2007)
10. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.-A.: An Algorithm for Conflict Resolution in Regulated Compound Activities. In: 7th Annual Int'l Workshop "Engineering Societies in the Agents World" (ESAW 2006), Dublin, Ireland (September 2006)
11. García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., Vasconcelos, W.: A Rule-based Approach to Norm-Oriented Programming of Electronic Institutions. ACM SIGecom Exchanges 5(5), 33–40 (2006)
12. García-Camino, A., Rodriguez-Aguilar, J.-A., Sierra, C., Vasconcelos, W.W.: A Distributed Architecture for Norm-Aware Agent Societies. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904. Springer, Heidelberg (2006)
13. Jaffar, J., Maher, M.J.: Constraint Logic Programming: A Survey. Journal of Logic Progr. 19(20), 503–581 (1994)
14. Jaffar, J., Maher, M.J., Marriott, K., Stuckey, P.J.: The Semantics of Constraint Logic Programs. Journal of Logic Progr 37(1-3), 1–46 (1998)
15. Kollingbaum, M.: Norm-governed Practical Reasoning Agents. PhD thesis, University of Aberdeen (2005)
16. Kollingbaum, M., Norman, T., Preece, A., Sleeman, D.: Norm Refinement: Informing the Re-negotiation of Contracts. In: Procs. Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN@ECAI 2006), Riva del Garda, Italy (August 2006)
17. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-Dimensional Dynamic Knowledge Representation. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173. Springer, Heidelberg (2001)
18. López y López, F.: Social Power and Norms: Impact on Agent Behaviour. PhD thesis, Univ. of Southampton (June 2003)
19. Norman, T., Preece, A., Chalmers, S., Jennings, N., Luck, M., Dang, V., Nguyen, T., Deora, V., Shao, J., Gray, W., Fiddian, N.: Agent-based Formation of Virtual Organisations. Knowledge Based Systems 17, 103–111 (2004)
20. Pacheco, O., Carmo, J.: A Role Based Model for the Normative Specification of Organized Collective Agency and Agents Interaction. Autonomous Agents and Multi-Agent Systems 6(2), 145–184 (2003)
21. Ross, A.: On Law and Justice. Stevens & Sons (1958)
22. Sartor, G.: The Structure of Norm Conditions and Nonmonotonic Reasoning in Law. In: Procs. 3rd Int'l Conf. on A.I. & Law (ICAIL1991), Oxford, England (July 1991)
23. Sartor, G.: A Simple Computational Model for Nonmonotonic and Adversarial Legal Reasoning. In: Procs. 4th Int'l Conf. on A.I. & Law (ICAIL1993). The Netherlands, Amsterdam (June 1993)
24. Sergot, M.: A Computational Theory of Normative Positions. ACM Trans. Comput. Logic 2(4), 581–622 (2001)

25. Swedish Institute of Computer Science. SICStus Prolog (viewed on 10 Feb 2005 at 18.16 GMT) (2005), http://www.sics.se/isl/sicstuswww/site/index.html
26. Vasconcelos, W., Kollingbaum, M., Norman, T., García-Camino, A.: Resolving Conflict and Inconsistency in Norm-Regulated Virtual Organizations. In: Procs. 6th Int'l Joint Conf. on Autonomous Agents & Multiagent Systems (AAMAS 2007), Honolulu, Hawai'i (2007)
27. White, S.M.: Requirements for Distributed Mission-Critical Decision Support Systems. In: Procs 13th Annual IEEE Int'l Symp. & Workshop on Eng. of Computer-Based Systs (ECBS 2006) (2006)

Alternative Dispute Resolution in Virtual Organizations

Jeremy Pitt, Daniel Ramirez-Cano, Lloyd Kamara, and Brendan Neville

Intelligent Systems & Networks Group, Dept. of Electrical & Electronic Engineering, Imperial College London, SW7 2BT, UK

Abstract. Networked systems are the driving force of modern business and commerce, underpinned by ideas such as agile enterprises, holonic manufacturing, and dynamic real-time supply chains. On occasions, the system operation will be sub-optimal or non-ideal, and disputes will occur between independent partners. It may be undesirable to resolve such disputes by recourse to law; preferably, the parties in dispute would settle the matter by themselves. Therefore, we develop an alternative dispute resolution (ADR) system for virtual organizations as a way of settling disputes internally. We provide a norm-governed specification of an ADR protocol which is, effectively, an intelligent agent-based autonomic system. We develop this specification in two ways: concretely, through description of the mechanisms underlying protocol operation; and abstractly, by considering how the specification addresses principles for jury trials.

1 Introduction

Networked systems are the driving force of modern business and commerce, underpinned by ideas such as:

- The agile enterprise: a decentralised, flexible and adaptive 'organization' which adjusts to changing environments or market opportunities with minimal disruption [1];
- Holonic manufacturing: coordination of factory components (or 'holons', i.e. machines, workbenches, plants, personnel, parts, etc.) for mass customization and accelerated product development [2]; and
- Real-time business intelligence: on-demand, responsive supply chains which use visible information to adapt to prevailing market conditions [3].

Characteristic features of all these ideas include decision-making based on local information, decentralised control, and heterogeneous — and therefore unpredictable — network components with potentially conflicting goals. As a result, there is scope for error: either by malice, disobedience, self-interest, accident, or necessity, system operation may occasionally deviate from the ideal. These are the same characteristics of *open agent societies* [4,5]. In previous work, we have proposed [6] a norm-governed approach to multi-agent systems, as a basis

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 72–89, 2008.

for specifying open agent societies which can, in turn, be used to realise agile enterprises, virtual organizations, and so on.

In this previous work, we have generally identified norm-violation and used some form of sanction mechanism to deal with it [6]. However, we also recognise that norm-violation can be subject to dispute between otherwise independent partners who are participating in an open system such as a virtual organization (VO) [7]. A commercial dispute in a 'real' organization would often be solved by time-consuming and costly litigation. Furthermore, taking a matter to court almost certainly damages the potential for any future business opportunities between the litigants. Therefore it may be undesirable to resolve such disputes by recourse to law: preferably, the parties in dispute would settle the matter by themselves. The savings in time, money and importantly long-term business relations suggest that amicable, mutually agreed, internal settlements are more in keeping with autonomic 'self-repair' mechanisms than adversarial, externally-decided, imposed 'crime and punishment' approaches to social order.

Accordingly, in this paper we present the basis of an alternative dispute resolution (ADR) system for virtual organizations as a way of settling disputes internally. We provide a norm-governed specification of an ADR protocol which is, effectively, an intelligent agent-based autonomic system supporting self-organisation and self-regulation in agent societies. Having described the mechanisms behind agent actions within the protocol, we then consider how the specification addresses principles for jury trials in human societies. We find that not only are these principles embodied by the specification, but that through them, we can also leverage massive scalability, 'cheap' communications and 'rapid' decision-making to achieve a pluralistic and representative computational society.

2 Background and Motivation

In this section we present in more detail the background and motivation for this work. We briefly consider a key driver for the research, the delivery of automated intelligent legal information systems; then, we review the background technology of norm-governed open multi-agent systems (open agent societies), and then consider the motivation for using ADR as a kind of autonomic system for virtual organizations.

2.1 Automated Legal Intelligent System

A *tripartite* system of government has, particularly through its legislature and judiciary, front-line responsibility for the underlying framework for economic activity in the governed society. Increasingly, the Internet and other communications networks are driving new models for this economic activity, including, as discussed, new models of B2B (Business-to-Business) commerce (i.e. agile enterprises, holonic manufacturing, etc.), but also new models of B2C and C2C (Business-to-Consumer and Consumer-to-Consumer) commerce, enabled by e-commerce, micro-payments and individually produced and distributed content.

It is partially the role of e-Government, then, with respect to the improvement of delivery of public services, to ensure that citizens' rights are respected in this new digital economy. Therefore, since it will inevitably be the case that legal conflict will arise, for example over contracts, payments, use of intellectual property, and so on, it is necessary to provide a mechanism for dealing with that conflict. In particular, a lightweight mechanism is required, to avoid expensive litigation over relatively modest sums, and unduly prolonged processes for what should, or could, be quickly resolved issues.

One objective of the EU ALIS project (IST 027968) is to develop a computational platform which offers legal information services that can provide exactly this lightweight mechanism. One underlying concept being used to develop this platform is the theory and technology of *norm-governed multi-agent systems*.

2.2 Norm-Governed Multi-agent Systems

A norm-governed multi-agent system can be expressed in terms of a set of *agents* (the members of a society), a set of *social constraints* on a society (*norms*, and other constraints, such as physical and logical constraints), a set of *roles* that members can play, the *state* of the members and the environment in which they act, a *communication language*, relationships between the members, including *ownership* and *representation* relations, and the *structure* of an open agent society (OAS). For more details, see [6].

In addition to the above features, we maintain the standard and long established distinction between physical capability, institutionalised power and permission (see, for instance, [8,9] for illustrations of this distinction). Accordingly, a specification of the social constraints of a norm-governed multi-agent system expresses four aspects of agent activity: (i) the physical capabilities; (ii) institutionalised powers; (iii) permissions, prohibitions and obligations of the agents; and (iv) the *sanctions* and *enforcement policies* that deal with the performance of forbidden actions and non-compliance with obligations.

Of particular interest in the current work is the refinement of the specification of sanctions and enforcement policies. What we seek is an identification of the norm-violation (the cause for sanction) or dispute over one of the other three aspects of agent behaviour, and an enforcement policy for 'repair' of the sanction or dispute that is based on Alternative Dispute Resolution.

2.3 Alternative Dispute Resolution

Alternative Dispute Resolution (ADR), especially on-line, is another option to litigation, or court resolution, and usually takes the form of negotiation, mediation or arbitration (cf. [10,11]). The rise in importance of ADR methods over the past two decades is due to the numerous benefits offered to the parties involved in a dispute (see below), coupled with the well known shortcomings of litigation. Consequently, investigating and implementing ADR in the context of intelligent legal system is an important focus of attention, and raises many key issues that citizens and businesses should consider when attempting to resolve a conflict.

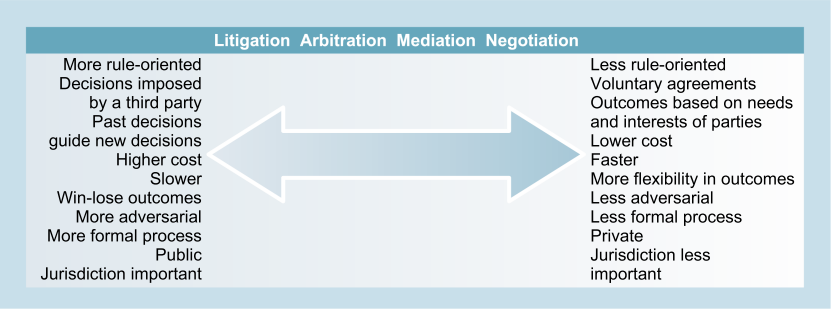


Fig. 1. Some relative attributes of dispute-resolution methods (from [13], Chart 7.2)

The problems of litigation have played a major part in the promotion of ADR to disputing parties, none more so than to firms, especially small to medium enterprises. Some reports indicate that very few lawsuits filed actually go to trial, and of this an even smaller proportion arrive at a verdict [12]. This is often due to a settlement being reached just prior to the end, or the case breaking down leading to a retrial. Equally, securing 'justice' for the individual citizen can be daunting when faced with the prospect of going to court, whereby the time, expense and hassle can easily outweigh the relatively small sums involved in the dispute. There are several initiatives to ensure that such minor cases can be resolved quickly, e.g. the Small Claims Court in the UK.

Even so, with many litigating cases reaching a settlement, they often come at a high price, in terms of money and time. In many cases, it can take years for a case to come to trial, time firms and individuals cannot afford to wait. During trial, companies and governments could be forced to wait for a prolonged period, impacting heavily on business. The significant capital expenditures make the litigation process expensive, with fees incurred for legal services, as well as the cost of court overheads. There is also the potential for appeals, which can immediately add to the mounting delay and cost.

By contrast, ADR carries numerous benefits [14], including often being opportune and relatively quick. It also allows the parties involved to have more control over their dispute, and so settlement, as they choose the procedure and terms and conditions. In addition, any third party required — for example in a mediation and arbitration procedure — can be determined by those involved, and can come from *within* the system, so that 'juries' are selected from a peer group with a genuinely shared experience, knowledge and understanding. Further distinctions between various types of dispute resolution appear in Fig. 1, which depicts a 'spectrum' of associated procedural qualities.

ADR and related approaches have also been seen as important auxiliary services to mainstream legal process. This is apparent in economic contexts — see for example, the Organisation for Economic Co-operation and Development's workshop on ADR [15] as well as the United Nations 2003 report on e-commerce [13] — and the assessments of a number of commercial entities [16, 17, 18]. For all of the above reasons, ADR is a particularly apposite

approach to building an 'autonomic' system for dealing with disputes and norm-violations in Virtual Organizations.

3 Alternative Dispute Resolution Protocol: Specification

In this section, we discuss the specification of an Alternative Dispute Resolution protocol. A full specification can be found in [19,20], including AUML diagrams specifying the sequence of actions in the protocol, and an *Event Calculus* (EC) [21] axiomatisation of those actions which determine the changing normative positions (i.e. in terms of power, permissions, sanctions, etc.).

There are in fact three methods for ADR in our system: *negotiation*, *mediation* (*negotiation* through a third party mediator), and *arbitration* (the dispute is put to a panel or 'jury' who adjudicate). In the following sub-sections, we concentrate on the specification of the *arbitration protocol* and the associated concepts of *arbitration panel composition*, *jury decision-making through opinion formation* and a *voting protocol*.

3.1 Arbitration Protocol

The arbitration protocol provides an impartial resolution to a dispute based on the expert legal opinion of a panel of arbitrators (strictly speaking, agents occupying the role of arbitrators). The resolution of the arbitrators is given in the form of an *award* which is final and not open to negotiation or appeal (unless specific provision has been made otherwise).

An important advantage of arbitration over litigation is that under the former, the disputants can choose which neutral third parties arbitrate the dispute and the rules of the session. Thus, a panel of arbitrators (the size of which is mutually agreed as a *norm* of the VO) is chosen from a potential pool of arbitrators. Membership of the pool can be open to any participant in the VO, or can be dependent on application-specific qualifications. For example, agents may learn domain-dependent legal process and strategies from participating in different mediation/arbitration cases. When a subsequent dispute resolution case arises, agents with the relevant domain knowledge and experience can be nominated for the pool from which the actual arbitrators for the case are selected.

The arbitrators consider all the relevant facts during the deliberative process and eventually formulate an award (binding on the parties) following a process of *opinion formation*. In this process each arbitrator can influence and be influenced by other agents to change its opinion.

The parties present their versions of the dispute and their demands. Each party has the opportunity to present any argument, evidence or document which might help their case. For the purpose of the current work it is assumed that a full ontology to represent relevant facts and laws exists and that the information presented by the parties is represented using the same ontology.

We propose the following (simplified) arbitration protocol for dispute resolution, inspired by the work of the WIPO arbitration and mediation center [18]:

(i) Arbitration panel composition; (ii) Voting, and (iii) Jury Decision-making through opinion formation. We now look at each stage in turn.

3.2 Arbitration Panel Composition

The selection of individual arbitrators depends on the preferences and strategies of the parties (who are each likely to choose arbitrators that are sympathetic to their respective case(s)). The panel selection process is, however, overseen centrally along the following lines:

- *Nomination*: Both parties nominate a list of arbitrators in order of preference.
- *Vetoing*: Both parties may veto some arbitrators from their counterpart's list.
- *Alternate strike*: Both parties successively remove one name from the list until the required panel size is met.
- *Chair appointing*: One arbitrator is appointed as chair, in addition to the arbitrators chosen by the panel (see Section 3.3).

The Event Calculus (EC) is an executable logic programming formalism for reasoning about time, action and events. This formalism can be used to both express and analyse the panel selection process. In an EC specification, this analysis determines both the multi-valued *fluents* (a state variable representing a system property whose value may change over time), and the actions which change the values of those fluents. Therefore the EC specifications consist primarily of axioms of the form:

$$\begin{aligned}
&\mathbf{pow}(L, nominate(L,C,N)) = true \ \ \mathsf{holdsat}\ \ T \leftarrow \\
&\qquad role_of(L) = litigator \ \ \mathsf{holdsat}\ \ T \ \ \wedge \\
&\qquad status_of(C) = agreed(T') \ \ \mathsf{holdsat}\ \ T \ \ \wedge \\
&\qquad method_for(C) = (arb, _) \ \ \mathsf{holdsat}\ \ T \ \ \wedge \\
&\qquad nominated(L,C) = [] \ \ \mathsf{holdsat}\ \ T
\end{aligned}$$

$$\begin{aligned}
&nominate(L1,C,N) \ \ \mathsf{initiates}\ \ status_of(C) = nominated \ \ \mathsf{at}\ \ T \leftarrow \\
&\qquad \mathbf{pow}(L1, nominate(L1,C,N)) = true \ \ \mathsf{holdsat}\ \ T \ \ \wedge \\
&\qquad length(N) > 5 \ \ \wedge \ \ length(N) < 13 \ \ \wedge \\
&\qquad role_of(L2) = litigator \ \ \mathsf{holdsat}\ \ T \ \ \wedge \\
&\qquad \mathbf{not} \ \ nominated(L2,C) = [] \ \ \mathsf{holdsat}\ \ T
\end{aligned}$$

The first axiom states that agent L is empowered to perform a nomination (of arbitrators), if it occupies the role of litigator, the status of the dispute C is that an ADR method has been agreed, and the method is *arb* (arbitration), and L has not already performed a nomination. The second axiom states that after performing a valid nomination, i.e. subject to being empowered and the constraints that the number of nominees is more than 5 and less than 13, the value of the status fluent of case C is *nominated*. Another axiom (not given here) sets the value of $nominated(L1, C) = N$.

3.3 Jury Decision-Making through Opinion Formation

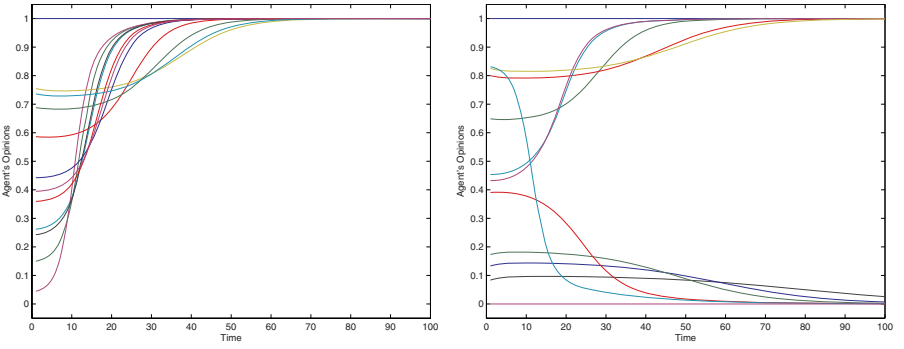
The decision-making process of a panel of arbitrators (i.e. the jury) can be modelled through a mathematical and logical formulation of an opinion formation dynamics [22]. One of the main characteristics of a jury is that arbitrators are distinct from one another and consequently, each arbitrator has a different mind-set, i.e. mental models of the law. In accordance with this characteristic, we introduce a numerical abstraction of an arbitrator's opinion on a specific case of award issuance. In addition, we provide a parameterisable set of formulas and associated algorithms for calculating and updating those opinions, where the parameters reflect the arbitrators' distinct initial mind-sets and the algorithms represent introspective and consultative deliberation by the arbitrators. A full account of the underlying mechanisms, as well as comparison with other opinion formation models, is given in [22]. The key elements for the purpose of the subsequent discussion are:

- *Topics*, the articles about which the arbitrators hold opinions. We treat topics here as being the specific cases — which, in turn, we consider to be synonymous with the facts of a case;
- *Opinions*, where an opinion is a function mapping from time-points to a number in the real number range $[0, 1]$. Each arbitrator has an opinion function whose instantaneous value reflects how much that arbitrator agrees with the current topic (0 representing total disagreement and 1 total agreement)[1];
- *Confidence*, a number in the range $[0, 1]$, with each arbitrator maintaining time-dependent confidence values for arbitrators of its acquaintance (or panel). The higher the confidence value for a particular associate, the more likely that arbitrator is to be influenced by the opinions of that associate. Arbitrators maintain confidence values for themselves, which capture the quality of *self-confidence*; this metric is treated the same way as confidence values for other arbitrators.

The process of opinion formation is guided by a designated chair arbitrator in accordance with the following principles. First, the arbitrators receive the facts surrounding a case and create an opinion about the award that should be issued, based on their respective initial mind-sets. The arbitrators then exchange opinions amongst themselves. According to individual opinion aggregating mechanisms and the confidence associated with the originator of each received opinion, arbitrators evaluate the given new opinions and — if influenced to do so — update theirs. If demonstrable consensus is reached following this exchange and update (i.e. if the subsequent round of opinion exchange shows all arbitrators to be of the same opinion) then an award can be issued. The arbitrators otherwise iteratively exchange their opinions until consensus is reached.

We omit further details on a number of aspects of the opinion formation procedure here. These include mechanisms for appointing a chair, initial opinion

[1] This formulation of an opinion assumes that only one topic, or case, is under consideration by an arbitrator at any time. Introduction of an additional parameter (the topic) allows arbitrators to deliberate upon multiple topics simultaneously.



(a) A Jury reaching concensus (b) A Jury with polarised opinions

Fig. 2. Example Jury Opinion Formation Outcomes

creation and opinion aggregation. This is because we consider these aspects to be configurable — and perhaps uncontrollable — aspects of the problem domain. For example, the selection of an arbitrator as chair may be on a strict rotation basis, random or by external (higher authority) appointment. Similarly, the means by which an arbitrator forms an initial opinion and performs subsequent opinion aggregation may only be known to its owner or designer. These considerations make the opinion formation procedure an ideal candidate for controlled, experimental settings in which to model and investigate configurations reflecting aspects of the corresponding real-world domain [22].

Fig. 2a and Fig. 2b illustrate two possible outcomes of the opinion formation procedure. Both graphs depict the variation of the opinions of a panel of arbitrators over time, each line corresponding to the opinion of a distinct arbitrator. In Fig. 2a, all the arbitrators eventually arrive at the same opinion about the award to be issued and thus consensus is reached. It is also possible, however, that the opinions will dichotomise (as in Fig. 2b), whereupon a *hung jury* results. It might be the case, for example, that some arbitrators have high self-confidence in their individually distinct opinions and therefore resist updating them. We have identified a number of possible resolutions to such deadlock situations, influenced by real-world solutions under similar circumstances. These include (i) empowering the chair arbitrator to make the final (*casting*) decision; (ii) requesting additional expert arbitrators to join the panel and the deliberation process; (iii) requesting additional evidence (further facts) from the parties; or (iv) initiating a law-making procedure in recognition of a novel or revealing case. Although these methods of resolution are not validated at present, we note that they can be expressed through the same specification technique appearing in Section 3.2. We plan to investigate the relative efficacy of such methods in future work.

3.4 Alternative Mechanisms of Jury Decision-Making

In jury decision-making the goal is to reach a consensus about what the verdict should be. Ideally, when the facts are clear and shared by every member of

the panel, the conclusion should easily be inferred. However empirical evidence collected from several real cases of jury decision-making shows that different factors influence jurors during the deliberation process. Variables such as panel size, panel instructions, strength of the evidence, initial juror preferences, interpersonal influence, definition of key legal terms, polling mechanics, etc. affect the decision-making process [23].

In Section 3.3 we have adopted opinion formation as a mechanism for jury deliberation. The field of opinion formation traditionally aims at finding an opinion aggregation mechanism which best simulates opinion change in real social groups. This simulations provide an insight into how groups reach consensus or how opinions fragment and polarise. Analysis is focused on the aggregation mechanisms used (e.g. majority rule, weighted average, etc.). However other variables such as the topology of the social network and the roles of the agents significantly affect the opinion dynamics. By making some assumptions about the social group we are able to simplify and adapt opinion formation as part of the process of jury deliberation (the other part of the process being the voting — or polling — stage through the voting protocol). Modeling through opinion formation has the flexibility to seamlessly accommodate some of the variables of the deliberation process such as panel size, initial juror preferences and interpersonal influence.

Similar approaches to jury decision making are *information integration models*, *Bayesian models*, *Poisson models* and *sequential weighing models* [24], the latter being the closest to our opinion formation approach. These models adopt psychological and mathematical concepts which give them the characteristic of being quantifiable and verifiable. However, they do not provide a fully accurate model of the decision-making process since they cannot account for the subjective factors which are part of cognitive processes [23]. A different, more cognitive-focused but less quantitative *'storytelling'* approach has been proposed [25,23] in which the jurors assemble and organise the information (e.g. evidence, facts, etc) of the case in the form of coherent story.

While many jury behaviours such as persuasive influence, faction size, majority persuasion and hung juries can be analysed using opinion formation, it is limited in modeling those concerned with reasoning and arguing about the facts of the case. Furthermore, none of the approaches previously described seems to be able to tackle this shortcoming. However, a different but well-studied approach to dealing with dialogical reasoning is that provided by argumentation theory.

In the field of computer science, argumentation has traditionally dealt with situations where conflict of opinions exist (Bench-Capon and Dunne [26], and Chesñevar et al [27] provide comprehensive reviews on argumentation, while Walker [28] considers the role of argumentation in ADR). Of special relevance to our study is the use of argumentation for legal reasoning [29], [30]. In particular, a jury making-decision process can benefit from the possibility of defining a system for defeasible and persuasive argumentation which would allow the use of notions such as argument (and their status, e.g. justified, overruled and defensible), counterargument, rebuttal and defeat.

In contrast with the previously described quantitative decision-making mechanisms, argumentation is not concerned with the process of aggregating opinions but defining arguments and assessing their validity. Argumentation also approaches the problem of clearly defining key legal terms (e.g. for the benefit of the jurors) since it incorporates an underlying logical language. Furthermore, the jury deliberation process can be specified as an argumentation protocol which outlines the procedural and deliberation characteristics of a jury-decision making process.

However, by adopting argumentation over opinion formation we would lose some of the quantitative and predictive abilities that the mathematical formulation behind opinion formation provide. Also, argumentation accounts for the persuasive ability of the arguments but not for that of the individuals. Opinion formation allows the jurors to assigns individual confidence values to the other jurors according to matching preferences, experiences, attitudes and values. Thus, it is suggested that a comprehensive juror decision-making specification should consider a complementarity approach integrating the advantages and attributes of argumentation, opinion formation and a voting protocol.

A combined approach would adopt an argumentation protocol such as the executable specification presented in [31]. This protocol already considers the physical capabilities of the agents, the rules of the protocol and the permissions, prohibitions and obligations of the agents. Additionally, the protocol should also include jury instructions, polling procedures (e.g. secret ballot or public vote) and the jury decision rule (e.g. unanimous decision, majority rule or two-thirds majority). On the other hand opinion formation mechanisms would be used as preference criterion between competing arguments by considering the interpersonal influences between jurors. Moreover, analysis of parameters related to procedural and deliberation characteristics such as jury size, initial juror preferences, faction size and faction shifts could still be done by observing their effect in the collective opinion dynamics.

3.5 Voting Protocol

Once the chosen arbitrators have individually considered the details of the case(s), they go through a formal voting process to establish the dispute outcome. This voting procedure is presented in [32], in terms of a formal characterisation of the powers, permissions and obligations associated with the roles of (for example) *chair* and *subject*. This formalisation can be used to 'enforce' correct declaration of the voting result.

4 Principles of Juries

In this section, we provide a (preliminary) analysis of our alternative dispute resolution service (ADR-S), and especially its provision for settling disputes by arbitration, with the 19 *Principles for Juries and Jury Trials*, as set down by the American Bar Association (ABA) [33]. These principles define the fundamental properties for the management of the jury system. In their words, *"Each principle is designed to express the best of current-day jury practice in light of existing*

legal and practical constraints". Therefore, it is appropriate to cross-reference the arbitration method of ADR-S to see if it adequately reflects this best practice. Furthermore, the arbitration method is 'close' to a jury trial; if we can show that the formal specification of the arbitration method respects the principles of juries and jury trials we have some indicators that our ADR specification is, in some sense, 'fair' and 'trustworthy'.

The 19 principles are divided into five groups according to the area of concern. These groups are:

- General principles;
- Assembling a jury;
- Conducting a jury trial;
- Jury deliberations; and
- Post-verdict activity.

In this paper, we only consider the eight principles in the general principles section, as the others are more specifically concerned with activity in human concerns and spaces (e.g. courtrooms) rather than the electronic agent environment with which we are concerned.

The presentation of the principles is given a single sentence summary, and then qualified by a number of explanatory clauses and sub-clauses. We shall refer to these caveats as 'the details'. We shall now look at each principle's summary in turn.

Principle 1. The right to jury trial shall be preserved
An informal characterisation of a right is to say that A has a right to (perform) X if A is empowered to perform X and no-one else is permitted or empowered to prevent it. So we need to ensure that the following domain constraint holds at all time points T:

$$\begin{aligned}
&\mathsf{happens}(\mathit{serve_writ}(L1, L2, C)) \ \mathsf{at} \ T' \wedge \\
&T' < T \ \wedge \\
&\mathit{status_of}(C) = \mathit{open4proposals} \ \mathsf{holdsat} \ T \\
&\leftrightarrow \\
&\mathbf{pow}(L2, \mathit{propose}(L2, C, \mathit{arb})) \ \mathsf{holdsat} \ T
\end{aligned}$$

Now, from the axioms defined in the previous section, if *L2* does indeed perform the *propose* action for arbitration, for which it is always empowered, then *L1* will not be permitted to reject it. In fact, this works for *L1* too: it can serve the writ and immediately propose arbitration; *L2* is not permitted to reject this either. In this way arbitration takes precedence over negotiation and mediations; moreover we can see that the right to a jury trial is indeed preserved.

Principle 2. Citizens have the right to participate in jury service and their service should be facilitated
This principle is slightly removed from the arbitration method *per se*. The facilitation is supported by the assumption, mentioned above, that the list of potential arbitrators (jurors, or panel members), is made freely available. It is

then necessary to ensure that those members of the community who qualify in the appropriate way are added to this list, and this list is freely accessible.

Then, the right to participate is supported, when the list of nominees for the arbitration panel is made and the alternative strike process is concluded, the ADR-S takes the following action:

$$\begin{aligned} \mathit{confirm\_panel}(\mathit{ALIS}, C) \ \mathsf{initiates}\ \mathit{role\_of}(J, C) = \mathit{juror}\ \mathsf{at}\ T \leftarrow \\ \mathit{status\_of}(C) = \mathit{struckout}\ \mathsf{holdsat}\ T \ \wedge \\ \mathit{nominated}(\_, C) = N\ \mathsf{holdsat}\ T \ \wedge \\ \mathit{member}(J, N) = \mathit{true}\ \mathsf{holdsat}\ T \ \wedge \\ \mathit{qualifies}(J, \mathit{juror}) = \mathit{true}\ \mathsf{holdsat}\ T \end{aligned}$$

Note that the last condition we would expect to trivially hold: the condition of qualifying to be a juror should hold in order to be on the ADR-S list of potential panel members, and all nominations should come from this list. Thus 'citizens' are empowered to act as jurors and there is not an action that can prevent or remove this right under normal circumstances. We retain the option for removal from the list (through malfeasance, poor performance, and so on), just as the ABA principles allow for disqualification for jury service if, for example, a person has been convicted of a felony.

Principle 3. Juries should have 12 members
The 'should' here is qualified by the details, and literally it means 'ideally' 12 and then stipulates no fewer than 6 under any circumstances.

Under the stricter (deontic) reading of 'should', by inspection, we can see that the process of nomination, veto/replacement, and strike out results in an arbitration panel of size 12. In fact, we can actually do rather better than this, and, given a Ccalc (which is an AI action language like Event Calculus, but can handle planning queries of this sort [34]) representation of the specification, we can *prove* that the required property holds using the following query:

Query. Given the initial state of the Alternative Dispute Resolution, is it possible to reach a state where within 35 transitions *status_of(C) = struckout* holds and if $\mathit{nominated}(L1, C) = N1$, $\mathit{nominated}(L2, C) = N2$, and $L1 \neq L2$, then $N1 + N2 > 12$?

Proof (sketch). The longest sequence of exchanged messages is 35 (comprising 1 serve writ, 6 proposals (2 propose-reject pairs and 1 propose-agree pair), 2 nominations (1 each), 12 for vetoes (maximum 3 each, each being a veto-*nominate1* pair) plus 2 finish-vetoes, and 12 for alternative strike. If the solver fails to find a state within 35 messages, we infer there is no state where the status of C is *struckout* and the cumulative size of the nominations by the litigants in the case is 12.

More generally, it can be asked why juries in computational societies 'should' have 12 members. The earliest traditions of judicial procedure invoking jury decisions, dating back to Athenian times, consisted of 1,000 to 1,500 'dikaste',

or citizens, while it was King Henry 2nd who introduced juries of 12 'free men' to stand in judgement on land disputes.

As a result, traditionally juries have consisted of 12 members, but court rulings in the US, for example, are reflected in the ABA principles that 12 is not essential, but a minimum of 6 is (in particular for criminal cases). The main basis for such reasoning is that 6 members is sufficiently large to be representative of the population, facilitates group deliberation, and avoids both internal and external attempts at intimidation or influence; while promoting the possibility of reaching a unanimous decision in a 'reasonable' span of time.

It is not unreasonable though, to suggest that in *online* alternative dispute resolution, we could actually go back towards the 'original' Athenian model. The values of 6 (nominations each) and 12 (total panel size) are simply parameters, and they can be changed at design-time, and even *at run-time by the agents themselves*, using the techniques proposed in [35]. We could then envisage circumstances when the arbitration panel is composed of the entire community. This is especially apposite in fully-automated environments, such as Virtual Organizations, where we could enjoy a "fully participatory democracy" where not only plebiscites and referenda would be commonplace but all disputes are settled by "the popular will". However, such possibilities could be resolved in peer-to-peer networks and e-commerce sites, where disputes between people over minor transactions or IPR infringements could be readily settled by reversion to the entire user community. In this way we could revert to the original Athenian model and rely on the 'wisdom of crowds' [36] to ensure fair hearings and just decisions.

Principle 4. Jury decisions should be unanimous
It is the provisions of the voting protocol, as reported in [32] that uphold this principle, rather than any of the axioms reported here. However, the voting protocol is embedded in the ADR-S, so the required property is preserved.

The key axiom in [32] is this one, which imposes on the chair the obligation to declare the result in the appropriate way in the case of majority voting, where the number of votes for (F) exceeds the number of votes against (A):

$$\begin{aligned}\mathbf{obl}(C, \mathit{declare}(C, M, \mathit{carried})) = \mathit{true}\ \ \mathsf{holdsat}\ \ T \leftarrow\\ \mathit{role\_of}(C, \mathit{chair}) = \mathit{true}\ \ \mathsf{holdsat}\ \ T\ \ \wedge\\ \mathit{status}(M) = \mathit{voted}\ \ \mathsf{holdsat}\ \ T\ \ \wedge\\ \mathit{votes}(M) = (F, A)\ \ \mathsf{holdsat}\ \ T\ \ \wedge\\ F > A\end{aligned}$$

If we demand unanimous decisions, we simply need to change the code thus:

$$\begin{aligned}\mathbf{obl}(C, \mathit{declare}(C, M, \mathit{carried})) = \mathit{true}\ \ \mathsf{holdsat}\ \ T \leftarrow\\ \mathit{role\_of}(C, \mathit{chair}) = \mathit{true}\ \ \mathsf{holdsat}\ \ T\ \ \wedge\\ \mathit{status}(M) = \mathit{voted}\ \ \mathsf{holdsat}\ \ T\ \ \wedge\\ \mathit{votes}(M) = (F, 0)\ \ \mathsf{holdsat}\ \ T\end{aligned}$$

Note that the 'should' again means 'ideally' rather than 'universally' and there are provisions in the details to allow for non-unanimous decisions. Again, we have the flexibility to modify the rules to incorporate these nuances, for non-unanimous decisions of juries between 6 and 12 members, or if we chose to expand the jury membership in the Athenian model, where reaching a unanimous decision with 1,000 jury members is clearly impractical.

Principle 5. It is the duty of the court to enforce and protect the rights to jury trial and jury service

There is no such thing as a 'court' within the ALIS platform in which our ADR-S operates, but the interpretation of the principle in the current context suggests that the responsibility for administration and maintenance of the arbitration dispute resolution method *within ALIS* remains exclusively with the designers, implementors and maintainers.

There are two aspects to this. One is when the ADR-S is being used to resolve disputes between automated agents. Here we have to ensure that the integrity of the 'rules which allow the rules to be changed', and the list of potential arbitrators, is preserved. This is beyond the scope of the present discussion.

The other aspect is when the agents are proxies and the ADR system is being used to resolve disputes between people in computer-mediated human-human interaction. What we have here is the human reliance of an ALIS ADR-S user on software to enact a decision correctly, no matter that some of the inputs determining that decision may be entered by other users. Following Reynolds and Picard [37], who studied the issue of privacy in affective computing, we propose to ground those decisions on mutual agreement. The form of this agreement is a contract.

Contractualism is the term used to describe philosophical theory that grounds morality, duty, or justice on a contract, often referred to as a 'social contract' [38]. Reynolds and Picard extend this notion to Design Contractualism, whereby a designer makes a number of moral or ethical judgements and encodes them, more or less explicitly, in the system or technology. The more explicit the contract, the easier it is for the user to make an assessment of the designer's intentions and ethical decisions. There are already a number of examples of (implicit and explicit) design contractualism in software systems engineering, including copyleft, TRUSTe, the ACM code of conduct, shareware, and so on,

When designing an intelligent legal system to provide decision support for juries or arbitration panels, the ADR-S system designers will need to make the procedures for jury selection and service clear, have rigorously transparent reporting and administration channels, and collect, analyse and publish information regarding the performance of the jury system and the jurors themselves. The requirement to present this information needs to be coded in a contract, and in the case of disputes concerning economic activity, we suggest the terms and conditions should be explicit, and act as a 'seal of authenticity' that the dispute resolution methods used are of an appropriate standard and rigour. Whether a user or agent 'signs up' to the terms and condition will depend on how well the designers' perceptions match the user's expectations.

Principle 6. Courts should educate jurors regarding the essential aspects of a jury trial
Much of the detailed provisions in this principle concern the orientation of people selected for jury service, and the use of plain language for jurors to understand their role and responsibilities. While this highlights the need, we can reasonably presume that such concerns are primarily addressed by the specification of appropriate ontologies, protocols, and so on.

The interesting provision, however, is the obligation upon jurors to refrain from discussing the case outside the jury room, and whether or not it is permissible to discuss the evidence amongst themselves. While it is an issue to be addressed in the implementation — and since it is undesirable to monitor all communications between jurors, it remains an open question how to ensure this — we note that this obligation (to refrain from) and permission essentially refer to a *physical* act rather than a speech act (relating to institutional facts) and therefore outside the scope of the Event Calculus axioms. Performing such a physical act, if reported or discovered, would lead to an institutional fact, i.e. that some sanction would need to be applied, for example, that the individuals concerned are no longer qualified to occupy the role of a juror.

Principle 7. Courts should respect juror privacy insofar as consistent with the requirements of justice and the public interest
This principle is mostly concerned with juror "voir dire", which is the process by which jurors are selected, or more often rejected, to hear a case. In the ADR-S, the process of the selection of the arbitrators is controlled by the nomination and alternative strike steps. If there is no interaction between the nominees and the parties involved in the dispute, then this principle is not of concern. However, if there is, then there are constraints that need to be placed on the limits of interaction (i.e. what are acceptable questions, etc.).

Principle 8. Individuals selected to serve on a jury have an ongoing interest in completing their service
The ABA principles suggest that jurors should only be removed for "compelling reasons". Equally, we should offer an incentive to arbitrators to complete the job to the best that they are able. While offering an incentive to software is not quite what we had in mind, the idea of a rational agent maximising its utility is a well-known notion. Therefore, we could imagine that it should be the goal of an arbitrator to reach an impartial judgement, and to be proficient in its assessments, which could be related to a reputation system.

Principles 9-19. The other principles are more directly concerned with human trials and are not so much of concern to an electronic alternative dispute resolution service. Some of the provisions further refine the selection process, others are concerned with ensuring that the jurors understand the applicable law. The latter is of particular concern, of course, but is beyond the scope of the present paper.

5 Summary and Conclusions

In this paper, we have considered the specification of an Alternative Dispute Resolution (ADR) protocol for dealing with exceptions (norm-violations and disputes) in norm-governed Virtual Organizations. We have described the advantages of such a protocol over more antagonistic, time-consuming and expensive methods of reaching settlements. We have also provided examples of the protocol specification in Event Calculus executable form, thereby demonstrating the basis for its automation, and, subject to extension through the use of additional action languages, verification and validation. We have identified operational elements behind the ADR protocol — in particular, arbitration and voting procedures, as well as the key role of opinion formation in establishing consensus among a panel of arbitrators. A particularly interesting feature of the arbitration procedure so formulated is the extent to which the principles of juries and jury trials are preserved.

We also consider the convergence of agents possessing different specialisations with intelligent opinion formation strategies and algorithms to have many beneficial characteristics, some of which we suggest below:

- *Robustness*: The ADR protocol does not rely on one 'judge' arbitrator which could be targeted for manipulation. It instead relies upon the knowledge and opinions of expert agents which enforce their expertise with every case. Moreover, compromised arbitrators can be replaced from the pool of available arbitrators without major modifications to the system
- *Reliability*: Since the information is distributed between several arbitrators, the ADR protocol is resilient to single points of failure. Additionally inconsistencies or contradictions in law can be more easily overcome as the agents will be designed to reach a consensus about how that law has been interpreted before, based on their past experiences.
- *Flexibility*: Instead of training each new agent when a new legislation is passed, a new arbitrator can be created with this new law and called to the relevant cases. All other arbitrators in the case can then learn from interacting with this new agent and add the new legislation to their knowledge base (as described in Section 3.3).
- *Multi-legislative*: since arbitrators in norm-governed Virtual Organizations specialise according to cases, they can also specialise according to the legislation in which they act. Thus it is possible to have expert arbitrators in domestic laws, European legislations and international treaties.

With these characteristics, we believe that it is possible to leverage massive scalability, 'cheap' communications and 'rapid' decision-making to achieve a remarkable degree of pluralism and representativeness in a computational society.

We are building systems in which to further investigate and validate the claims made in this paper. We have already constructed one such system which integrates the ADR protocol with intelligent strategies for decision-making and opinion formation about frequently occurring issues in commercial disputes. Preliminary results suggest that it is indeed possible to build ADR into a norm-governed multi-agent system and subsequently resolve conflicts efficiently and

effectively with minimal cost to the parties involved. Future work will involve extending, testing and analysing the approach in different settings. One particular setting is the ALIS project, in which such approaches are required for automated conflict resolution for a large number and variety of legal cases.

Acknowledgements

We thank Sandip Dhillon for his work on the implementation referred to in Section 5. We are also extremely grateful for his contributions to an earlier revision of this paper and to the reviewers for their insightful and helpful comments. This work has been supported by the EU ALIS project (IST 027968).

References

1. Brafman, O., Beckstrom, R.: The Starfish And The Spider. Penguin (2006)
2. Mařík, V., William Brennan, R., Pěchouček, M. (eds.): HoloMAS 2005. LNCS (LNAI), vol. 3593. Springer, Heidelberg (2005)
3. Boyson, S., Corsi, T.: Managing the real-time supply chain. In: HICSS 2002, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2002)
4. Pitt, J., Mamdani, A., Charlton, P.: The open agent society and its enemies: a position statement and research programme. Telematics and Informatics 18(1), 67–87 (2001)
5. Pitt, J.: The open agent society as a platform for the user-friendly information society. AI Soc. 19(2), 123–158 (2005)
6. Artikis, A., Sergot, M., Pitt, J.: Specifying norm-governed computational societies. ACM Transactions on Computational Logic (to appear)
7. Cevenini, C.: Legal considerations on the use of software agents in virtual enterprises. In: Bing, J., Sartor, G. (eds.) The Law of Electronic Agents, vol. 4, pp. 133–146. Unipubskriftserier, Oslo (2003)
8. Makinson, D.: On the formal representation of rights relations. Journal of Philosophical Logic 15, 403–425 (1986)
9. Jones, A., Sergot, M.: A formal characterisation of institutionalised power. Journal of the IGPL 4(3), 429–445 (1996)
10. Schultz, T., Kaufmann-Kohler, G., Langer, D., Bonnet, V.: Online dispute resolution: The state of the art and the issues. Technical report, Report of the E-Com / E-Law Research Project of the University of Geneva (2001)
11. Slate II, W.: Online dispute resolution: Click here to settle your dispute. Dispute Resolution Journal 8 (2002)
12. WIPO Arbitration and Mediation Center: Dispute resolution for the 21st century (2007), http://arbiter.wipo.int
13. United Nations. In: United Nations Conference on Trade and Development — E-commerce and development report, ch.7, New York, Geneva, vol. UNCTAD/SIDTE/ECB/2003/, pp. 177–203 (2003), http://www.unctad.org/en/docs/ecdr2003ch7_en.pdf
14. Kowalchyk, A.W.: Resolving intellectual property disputes outside of court: Using ADR to take control of your case. Dispute Resolution Journal 61(2), 28–37 (2006)
15. OECD: OECD workshop on dispute resolution and redress in the global marketplace: Report of the workshop. Technical Report DSTI/CP(2005)9, Organisation for Economic Co-operation and Development (2005)

16. Ware, S.J.: Principles of Alternative Dispute Resolution. West Group (2007)
17. American Arbitration Association (2007), http://www.adr.org/drs
18. WIPO Arbitration and Mediation Center: Guide to WIPO Arbitration (2007), http://arbiter.wipo.int
19. ALIS: Deliverable D3.1: Formal characteristics of legal and regulatory reasoning from the computational logic point of view. Available from ISN Group, EEE Dept., Imperial College London (2008)
20. ALIS: Deliverable D3.2: ALIS ADR-S: The ALIS alternative dispute resolution service. Available from ISN Group, EEE Dept., Imperial College London (2008)
21. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Generation Computing 4(1), 67–96 (1986)
22. Ramirez-Cano, D., Pitt, J.: Follow the leader: Profiling agents in an opinion formation model of dynamic confidence and individual mind-sets. In: IAT, pp. 660–667 (2006)
23. Devine, D.J., Clayton, L.D., Dunford, B.B., Seying, R., Pryce, J.: Jury decision making: 45 years of empirical research on deliberating groups. Psychology, Public Policy, and Law 7(3), 622–727 (2001)
24. Pennington, N., Hastie, R.: Juror decision-making models: The generalization gap. Psychological Bulletin 89(2), 246–287 (1981)
25. Bennett, W.L.: Storytelling in criminal trials: A model of social judgment. Quarterly Journal of Speech 64(1), 1–22 (1978)
26. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. 171(10-15), 619–641 (2007)
27. Chesñevar, C.I., Maguitman, A.G., Loui, R.P.: Logical models of argument. ACM Comput. Surv. 32(4), 337–383 (2000)
28. Walker, G.B., Daniels, S.E.: Argument and alternative dispute resolution systems. Argumentation 9, 693–704 (1995)
29. Sartor, G.: A formal model of legal argumentation. Ratio Juris 7(2), 177–211 (1994)
30. Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. Artificial Intelligence and Law 4(3-4), 331–368 (1996)
31. Artikis, A., Sergot, M., Pitt, J.: An executable specification of a formal argumentation protocol. Artif. Intell. 171(10-15), 776–804 (2007)
32. Pitt, J., Kamara, L., Sergot, M., Artikis, A.: Voting in Multi-Agent Systems. The Computer Journal 49(2), 156–170 (2006)
33. American Bar Association: Principles for juries and jury trials (2005), http://www.abanet.org/juryprojectstandards/principles.pdf
34. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artif. Intell. 153(1-2), 49–104 (2004)
35. Kaponis, D., Pitt, J.: Dynamic specifications in normative computational societies. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 265–283. Springer, Heidelberg (2007)
36. Surowiecki, J.: The wisdom of crowds. Doubleday (2004)
37. Reynolds, C., Picard, R.: Affective sensors, privacy, and ethical contracts. In: CHI 2004: extended abstracts on Human factors in computing systems, pp. 1103–1106. ACM Press, New York (2004)
38. Rawls, J.: A Theory of Justice. Belknap Press (1999)

Electronic Institutions Infrastructure for e-Chartering

Manolis Sardis and George Vouros

University of the Aegean, Department of Information and Communication Systems Engineering, 83200 Karlovassi, Samos, Greece
{sardis,georgev}@aegean.gr

Abstract. The need of methodologies and software tools that ease the development of applications where distributed (human or software) agents search, trade and negotiate resources is great. On the other hand, electronic institutions of multiple agents can play a main role in the development of systems where normative specifications play a vital role. Electronic institutions define the rules of the game in agent societies, by fixing what agents are permitted and forbidden to do and under what circumstances. In this paper we present a case study on the use of specific tools, supporting the specification, analysis and execution of institutions for Maritime e-Chartering, proposing an infrastructure for Internet-based Virtual Chartering Markets (MAVCM).

Keywords: Electronic Institutions, Maritime e-Chartering, Multi agent systems.

1 Introduction

Electronic business and agents are among the most important and exciting areas of research and development in information and communication technology, with considerable potential impact and opportunities for the Maritime sector [35][36]. This paper proposes an infrastructure for Multi-Agent, Internet-based Virtual Chartering Markets (MAVCM). The MAVCM system aims to support business-to-business transactions in Maritime markets, providing mechanisms for Internet-based e-Chartering services. The proposed system offers a solution for efficiently handling the processes involving cargo owners (Charterers) and ship owners (Shipbrokers). Cargo owners aim to find ship owners to deliver cargoes at certain freight rates. The objective of MAVCM is to enable Maritime market participants to electronically charter and trade cargos, via their software agents.

The development of an e-Chartering system involving human and software agents is one of the most challenging applications for the Maritime domain of applications, due to the complexity of the task. Factors such as charter selection, port time, selection of route, costs of cargo handling, communications reliability and efficiency are critical and difficult to be combined in a detailed design of a highly distributed, open and dynamic multi-agent system.

In this paper, our aim is to analyze and extend the design specifications for this complicated system using Electronic Institutions (EIs) [7], supporting agents' collaborative activities. Furthermore, the paper proposes a solution towards the implementation of the proposed e-Chartering MAVCM electronic institutions infrastructure in terms of specific development methodologies and frameworks.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 90–107, 2008.

EIs are open systems that comprise autonomous, independent entities that must conform to common, explicit interaction conventions. The idea behind EIs is to mirror the roles traditional institutions play in the establishment of 'the rules of the game': The set of conventions that articulate human agent's interactions [7][11][27]. EIs specifications are both descriptive and prescriptive: The institution makes the conventions explicit to participants, and it warrants their compliance. EIs, as artifacts, involve a conceptual framework to describe agent interactions, as well as an engineering framework to specify and deploy actual interaction environments [24].

In this paper we specify how EI-related technology can contribute to solving the problem of agents' (Ship owners and Cargo owners) participation in different Maritime ports for the creation of valid offer/position combinations. As already pointed, our aim is the development of an e-Chartering infrastructure for MAVCM. Although our approach has things in common with some of the agent development methodologies and conceptual specifications of multi-agent systems proposals [28][33], the use of EIs contribute to the specification and development of the MAVCM system due to their following distinctive features:

- EIs allow the description of the roles and interactions of both human and software agents in a specific setting (the institution) using a comprehensive framework
- EIs make explicit the relationship between the computational framework developed by MAVCM and the existing maritime transactions for e-Chartering
- EIs clarify the difference between the behavior and the particular strategies the human and software agents may follow in pursuing individual goals

In this paper, we look into the EI artifact from a methodological perspective: we discuss the notions that underlie the conceptual EI framework and study the MAVCM system development process by means of existing ad-hoc software tools. Notably, many development tools have been proposed for the agent technologies [1][2][4], but each one has been proposed for a specific task and for only a part of the whole development cycle. By using the ISLANDER [6] electronic institutions editor, we describe the EI details that control the MAVCM agents' behavior. Finally, we propose development solutions for those parts of the MAVCM system that are incorporated into EIs [27].

The remainder of the paper is organized as follows. In section 2 we present the e-Chartering case study characteristics based on our previous work [3] and we clarify the problem. Sections 3 and 4 present the proposed solution using EIs and we discuss the methodology that constitutes the conceptual EI framework based on MAVCM design considerations and characteristics. The development considerations of EIs and the Electronic Institutions Development Environment (EIDE) framework for the development and testing of the complicated infrastructures for EIs in the MAVCM system are presented in section 5.

2 Actual World Institutions for the e-Chartering Case Study

In our case study we deal with a traditional chartering institution, where Ship owners and Cargo owners have to participate through this institution for reaching, in a best

price and under certain conditions and terms, an agreement for a contract for the transferring of specific cargo through sea. The whole system has been analyzed from its business perspective in [3]. In [3] we have used the GAIA [28] methodology and AUML [33][34] for the analysis and design of the core parts for the system.

The chartering domain involves an inherently distributed, open and dynamic environment, where agents are located in different geographical regions, new agents may arrive seeking for an offer or making a position, existing agents leave the setting, while unforeseen events may affect (via new opportunities or new threats) agents' participation. e-Chartering offers new possibilities to this domain: Parties, due to the potential of an e-chartering infrastructure to effectively tackle the above mentioned inherent problems/opportunities effectively, can benefit greatly to participate, react and reach effective solutions in such a setting. More specifically, the e-Chartering opens new roads for the Maritime community because it may involve multiple trading parties, balancing offers and positions, exploit awareness mechanisms for making participants aware of new opportunities and threats. Moreover, the online procedure, without the need for extra third parties, reduces the cost of the transactions. This, in conjunction to the fact that involved parties are facilitated to reach effective solutions (i.e. solutions with lower costs) in less time, gives the opportunity for small companies to participate in e-Chartering transactions, in cases where, in real human Maritime chartering, they could not afford to be involved.

The brokering procedure during e-Chartering can be described as a space where several *scenes* take place simultaneously, at different places, but with some causal continuity. Each scene involves various agents who at that moment perform well-defined functions. The scenes and their continuity are shown in a very sketchy way in Fig. 1. The first scene is the *registration scene*, where different Cargo owners and Ship owners have to register their characteristics and to inform the system for their offers/positions either for cargo or for vessels (Fig. 1). In this scene each Cargo owner submits in the system an 'order' and each Shipowner a 'position'. The order and the position are xml files containing the necessary detailed information for cargo and vessel characteristics, facilitating agents' reaching a contract agreement.

The content/attributes of orders and positions are shown in Fig. 2. The *investigation scene* involves MAVCM agents that search in their domain or in different

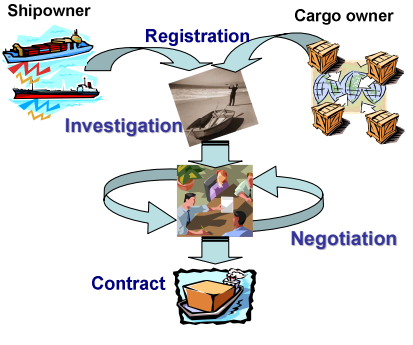


Fig. 1. e-Chartering scenes

Order	Position
-CHARTERER'S NAME AND DOMICILE -CARGO QUANTITY AND DESCRIPTION OF THE COMMODITY -LOADING AND DISCHARGING PORTS -THE PERIOD WITHIN WHICH THE VESSEL IS TO BE PRESENTED FOR LOADING (LAY/CAN) -LOADING AND DISCHARGING RATES AND TERMS -ANY RESTRICTIONS OR PREFERENCES REGARDING TYPE OR SIZE OF SHIP OR AGE OR FLAG -C/P FORM ON WHICH THE CHARTERER WISHES TO BASE THE TERMS AND CONDITIONS -COMMISIONS TO BE PAID BY THE OWNER	-SHIPOWNER'S NAME AND DOMICILE -DESCRIPTION OF THE VESSEL -THE PERIOD WITHIN WHICH THE VESSEL IS AVAILABLE -TYPE OF CHARTERING (VOYAGE, CONSECUTIVE VOYAGES, TIME, BAREBOAT, COA) -LOADING AND DISCHARGING RATES AND TERMS -ANY RESTRICTIONS OR PREFERENCES REGARDING CARGO -C/P FORM ON WHICH THE SHIPOWNER WISHES TO BASE THE TERMS AND CONDITIONS -COMMISIONS TO PE PAID BY THE OWNER

Fig. 2. Order & position elements

domains (i.e. ports), for finding a relevant position or an order, according to their position/offer. The *negotiation scene* follows the *investigation scene.* It involves agents performing contract negotiations based on end-user requirements. Finally, during the *contract scene* agents finalize or propose a contract between the Ship owner and the Cargo owner.

During the *investigation scene,* the agent that is responsible for a specific Ship owner (position) will search, based on user-specified search criteria, for a matching order. Figure 3 depicts a typical setting of two ports, each supported by an EI, involving Cargo owners and Ship owners (represented by agents) that pose orders and positions, respectively. These agents are acquaintances with broker agents that aim to coordinate agents' behavior. For example, Figure 3 shows a number of *broker* agents in ports A and B involved in the task. The search results from each broker agent must be analyzed and presented, so that all agents to be informed for the 'search solution', reaching a coherent and consistent solution: Specifically, a conflict in a port occurs when at least two brokers consider a different solution for the same pair or agents, or when at least one broker considers a solution involving different pairs with the same cargo owner or ship owner.

This problem can be extended in real-world cases where a large number of agents are involved in the same or in different ports with high Maritime chartering traffic. In this case, information from port policies (either for cargo like rules for chemical/toxic cargo, or for vessel rules like port pollution restrictions) and brokering conditions that should be followed and controlled by the *broker* agents constitute also sources of conflict. As it is shown in Figure 3, in this paper, we propose the use of electronic institutions in each port infrastructure: Different EIs operate in port A and port B. To reach coherent and consistent solutions among agents within and across several EIs we extend the EI infrastructure by introducing the *Broker Manager* agent as a coordinator of the *broker* agents within the same port and as a coordinator and communication interface between different EIs. This is further presented in the paragraphs that follow. The coordination between different EIs operating in different ports is not coordinated by a central authority as this would make the system to be centralized: We rather consider broker managers to constitute the backbone of the system structure, operating in a decentralized way.

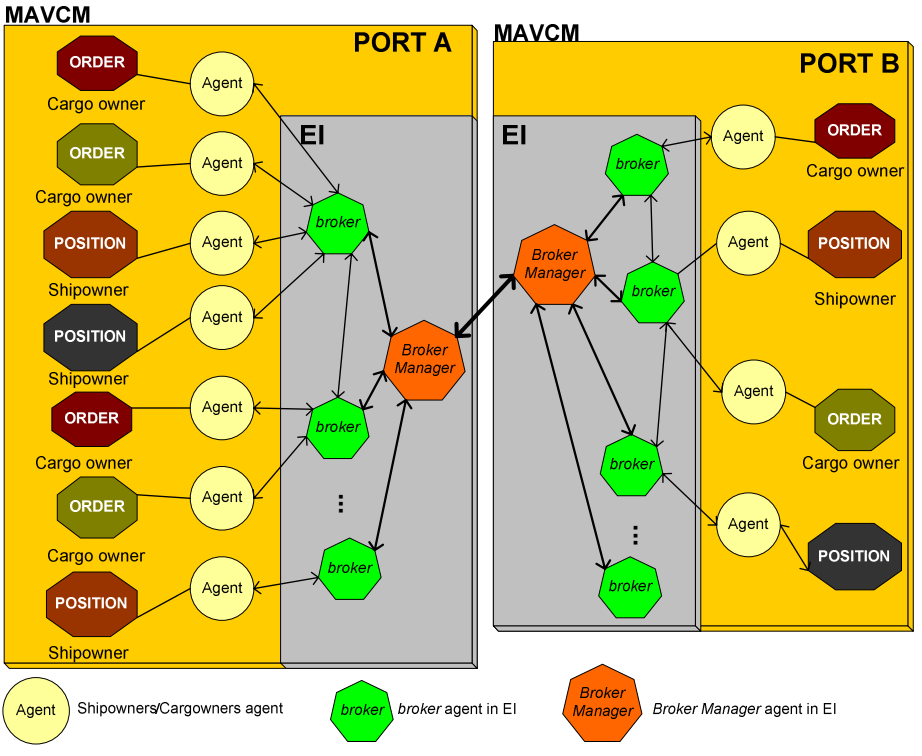


Fig. 3. The Electronic Institutions in e-Chartering process

It is clear that the *investigation* and the *negotiation scenes* are the core and the most complex scenes for the MAVCM system. This is so because the number of the participating agents in different, numerous ports and the number of actual messages between them increases exponential, based on Maritime requirements for world wide area business transactions. These scenes have been analyzed and designed using the EIs and the EIDE framework described in section 5.

3 Electronic Institutions Fundamental Concepts

The role of any formal method is to provide a clear and precise description of *what* a system is supposed to do, rather than a description of *how* it operates [5]. The presence of an underlying formal model supports the use of structured design techniques and formal analysis, facilitating development, composition and software reuse. Traditional institutions can be viewed as 'a set of artificial restrictions that articulate agent interactions'. Analogously, when looking at computer-mediated interactions we think of EIs as a regulated virtual environment where the relevant interactions among participating entities take place. The core notions of an electronic institution include:

- Agents and Roles. Agents are the players in an electronic institution, interacting by the exchange of illocutions. Roles are defined as standardized patterns of behavior. The identification and regulation of roles is considered to be part of the formalization process of any organization [13]. Any agent within an electronic institution is required to adopt some role(s). While dialogical schemata are associated to roles, an agent adopting a given role must perform the actions that instantiate the corresponding schemata
- Dialogical framework. The context of interaction amongst agents of an institution, such as the objects of the world and the language employed for communicating. Agents interact through illocutions using a common ontology (vocabulary), i.e. they use a common language for communication and they share the same conceptualization of their domain
- Scene. Interactions between agents are articulated through agent group meetings, which are called *scenes*, with a well-defined communication protocol. The protocol of a scene is considered to delineate the possible dialogues agents may have
- Performative structure. Scenes can be connected, composing a network of scenes. This captures the existing relationships among scenes. The specification of a performative structure contains a description of how the different roles can legally move from scene to scene. A performative structure is to contain the multiple, simultaneous ongoing activities, represented by scenes. Agents, within a performative structure, may participate in different scenes at the same time with different roles
- Normative rules. Agent's participation and further activity in the context of an institution may have consequences that either limit or enlarge its subsequent acting possibilities. Such consequences will impose obligations and/or rights to the agents and affect its possible paths within the performative structure

The agents, roles and scenes in e-Chartering have already been defined in detail [3], but with no collaboration between the involved *broker* agents. Section 4 of this article describes the dialogical framework, the performative structure and the normative rules of the EI.

4 Electronic Institutions Structure

As the development of electronic institutions (EIs) is highly complex and critical, this section briefly presents the constituent elements of EIs, as well as their design and development methodology [7][8]. In order to clarify the meaning of 'relevant interactions in a regulated environment' we present roles and the dialogical framework which allow us to express the syntactic aspects of EIs and the ontology of a particular EI. Then we present the elements that allow us to express the prescriptive aspects of EIs.

As already said, we have used the ISLANDER editor for expressing and designing in detail all the syntactic aspects of the e-Chartering EIs. This editor supports the institution designer with combined textual and graphical elements for the specification and verification of electronic institutions design. The output of the editor is an XML file that can be used from other Electronic Institutions Development Environment (EIDE) tools towards completing the development phase of the e-Chartering EIs.

4.1 Roles

In order to participate in an EI, an agent is obliged to adopt some role(s). Roles are finite set of dialogical actions that enable abstracting from the individuals in the activities of an institution. The agent that plays a role must conform to the pattern of behavior corresponding to that particular role. In human institutions, roles are usually hierarchically organized, and sometimes roles may produce conflicting results/effects. For example, turning into our case study, a *broker* agent may search for a proper position of a vessel based on a Cargo owner order (Cargo owner agent). At the same time another *broker* agent searches for an order based on the vessel position (Ship owner agent). Both *broker* agents may achieve the same result, given a particular vessel and a particular cargo. It should be noted that there is a limit on the manipulated number of external Ship owner and Cargo owner agents from each *broker* agent. The limit is based on Maritime chartering rules and brokering classifications [37]. This is a case where a role hierarchy can be effective for EIs performance and quick response on user requests, as higher-level roles may coordinate subordinate roles to reach consistent and coherent decisions.

As Figure 3 depicts, the most important involved roles in the e-Chartering framework are the *agents* representing Ship owners and Cargo owners, the *broker* agents that control one or more *agents*, and the *Broker Managers* that control one or more *broker* agents lower in the role hierarchy. Each Maritime port has a single *Broker Manager*. The *Broker Manager* is the most experienced member of a port, with respect to the e-Chartering rules and policies, and controls the final decisions during the investigation scene. It also supports the end user (Shipowner or Cargo owner) by suggesting, after an evaluation procedure and by using *External Resources* [3] agents' feedback, charters from the same port or from other ports. This is done by exchanging information with the other *Broker Manager* agents. The *Broker Manager* role works as an interface between different Maritime ports and is responsible for the communication, information exchange and ports' cooperation.

4.2 Dialogical Framework

The Dialogical Framework [10] is composed of a communication language, a representation language and an ontology. Agents are sharing this framework, so heterogeneous agents are able to exchange knowledge with one another. To clarify the available illocutions for agent dialogues in a given institution, a dialogical framework is defined [24] as a tuple:

$$DF = <O, L, I, R_I, R_E, R_S> \quad (1)$$

where,

- O stands for the EI domain ontology
- L stands for a content language to express the information exchanged between agents
- I is the set of illocutionary particles
- R_I is the set of internal roles
- R_E is the set of external roles
- R_S is the set of relationships over roles

The domain ontology involves orders and positions and their attributes, together with possible actions that agents must undertake during their participation in EIs.

Considering roles, we have to recall that, in a chartering procedure, a given agent may act as a buyer (Cargo owner) at some point, and as a seller (Ship owner) at another, and many agents may act as buyers. This consideration allows us to think of participants adopting *roles*. All agents adopting a given role should be guaranteed to have the same rights, duties and opportunities. We differentiate between the *internal* and the *external* roles. The *internal* roles define a set of roles that will be played by EI staff agents. In our case the *Broker Manager* and the *Broker* agent are internal roles. An *external* role in the MAVCM infrastructure is for example the *External Resources* agent that is responsible to bring into the EI external info (for example, market is very low and duration of charter is long, voyage passes through regions which are to be avoided by user choice) that is specific to Maritime charter brokering procedures during charter investigation. Since an EI delegates services and duties to the internal roles, an external agent (i.e. an agent playing an external role) is never allowed to play any of them. Finally, we need to define relationships among roles, specifying for instance roles that cannot be played at the very same time, or roles that have some authority over others. In this last case, we need to specify that the *Broker Manager* agent controls the *broker* agents' transactions and activities. On Figure 4, internal roles are displayed in light and external roles (*CargOwner*, *Shipowner*, *ExternalResources*) are displayed in dark.

The *Broker Manager* and the *Broker* have a static separation of duties relation (ssd), as e-Chartering agents cannot play both of these roles at the same time within the institution.

In the MAVCM infrastructure, each Maritime port chartering procedure is described by at least one EI (a port could have more than one EI, in order to support chartering traffic). Each EI is using a dialogical framework to support the involved agents with the type of illocutions exchanged during the e-Chartering *scenes*. The maritime brokering procedures determine the involved illocutions.

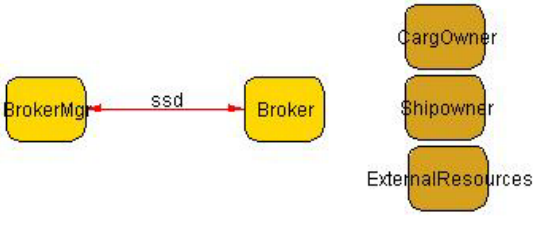


Fig. 4. Dialogical framework for e-Chartering (using ISLANDER editor)

4.3 Scene

Scenes [24] specify "group meetings" that articulate well defined communication protocols, modeling the dialogical interactions between roles. Scenes are graphically been represented using states and labeled arcs between them: Based on the scene context. Labels represent the transitions from state to state and impose restrictions on the paths that the scene execution can follow. Scenes allow agents either to enter or to

leave a scene at some particular states of an ongoing conversation. More formally a *scene* is a tuple [24]:

$$s = <R, DF_s, W, w_o, W_f, (WA_r)_{r\varepsilon R}, (WE_r)_{r\varepsilon R}, \Theta, \lambda, min, max> \quad (2)$$

where,

- R is the set of scene roles involved in that scene
- DF_s is the restriction to the scene *s* of the EI dialogical framework
- *W* is the non-empty set of scene states
- $w_o \varepsilon W$ is the initial state
- $W_f \subseteq W$ is the set of final states
- $(WA_r)_{r \in R} \subseteq W$ is a family of non-empty sets such that WA_r stands for the set of access states for role $r \in R$
- $(WE_r)_{r \in R} \subseteq W$ is a family of non-empty sets such that WE_r stands for the set of exit states for role $r \in R$
- $\Theta \subseteq WxW$ is a set of directed edges
- $\lambda: \Theta \rightarrow L$ is a labelling function, where *L* can be a timeout, an illocution schema or a list of constraints
- min, max : $R \longrightarrow N$ min(*r*) and max(*r*) specify the minimum and maximum number of agents that must and can play the role $r \in R$

The specification of scenes and their interconnection in performative structures are explained and depicted in the section that follows.

4.4 Performative Structure and Transitions

As in real life, during a chartering procedure, one vessel after a finished trip may search for a cargo, and after a while it will search again for a new cargo and so on. As a consequence, scenes are repeated (in periods of days or months). Likewise e-Chartering procedures, where some part of their functionality will be transformed and defined based on EIs, they follow a performative structure. This structure specifies the interlacing of regular scenes and transitions. Transitions specify the relationships among scenes. Complex activities/relations between scenes include:

- Causal dependencies
- Synchronization mechanisms
- Parallelism mechanisms
- Choice points that allow roles leaving a scene to choose their destination
- Role flow policy among scenes, specifying for example which paths can be followed by the roles leaving a scene and which target scenes they can reach

The graphical representation of the above stated activity can be based on AUML roles and graphic presentations. The performative structure becomes populated by agents that make it evolve whenever agents comply with the rules encoded by the specification. However, a single agent can possibly participate in multiple scenes at

the same time. Therefore there is a need for a formal specification of performative structures that is expressive enough to facilitate the specification of such rules. Scenes and transitions are connected by means of directed *arcs*. Labels on the directed arcs determine which agents, depending on their roles, can progress from scenes to transitions or from transitions to scenes. Transitions are divided into two parts: the *input*, "receiving" agents from the incoming arcs, and the *output*, through which agents leave following the outgoing arcs towards other scenes. In all EIs there is always an *initial* and a *final* scene, where are the entry and exit points of the institution. Technically, the definition of a performative structure is as follows:

$$PS = (S, T, s_o, s_\Omega, \mathrm{E}, f_L, f_T, f_E^O, C, ML, \mu) \tag{3}$$

where,

- S is a non-empty set of scenes
- T is a set of transitions
- $s_0 \in S$ is the initial root scene
- $s_\Omega \in S$ is the final output scene
- $E = E^I \cup E^O$ is a set of arc identifiers where $E^I \subseteq S \times T$ is a set of edges from exit states of scenes to transitions and $E^O \subseteq T \times S$ is a set of edges from transitions to scenes
- $f_L : E \rightarrow DNF_2 v_{AxR}$ maps each arc to a disjunctive normal form of pairs of agent variable and role identifier representing the arc label
- $f_T : T \rightarrow \Im$ maps each transition to its type
- $f_E^O : E^O \rightarrow E$ maps each arc to its type (one, some, all or new)
- $C : E \rightarrow ML$ maps each arc to a meta-language expression of type Boolean, i.e. a formula representing the arc's constraints that agents must satisfy to traverse the arc
- ML is a meta-language
- $\mu : S \rightarrow \{0,1\}$ states whether a scene can be multiply instantiated at run time or not

The classification of transitions in classes T= {and/and, or/or, and/or, or/and} is based on the behaviour that they exhibit on their input and output sides:

- *And/and* are establishing synchronization and parallelism points
- *Or/or* specify asynchronous way at the input and choice points at the output
- *And/or* synchronize agents on the input and they permit agents to individually make the choice of which path to follow when leaving
- *Or/and* specify that agents are not required to wait for others in the input side, but are forced to follow all the possible outgoing arcs

Finally, the set of arc types are E= {1, some, all, *}. *1*-arcs constraint agents to enter a single scene instance of the target scene, *some*-arc allows the agents to choose a subset of scene instances to enter, *all*-arc forces the agents to enter all the scenes instances, and finally the *-arc instantiates the creation of a new scene instance of the target scene. Using the ISLANDER tool from the EIDE framework [6][24] the e-Chartering performative structure is depicted in Figure 5.

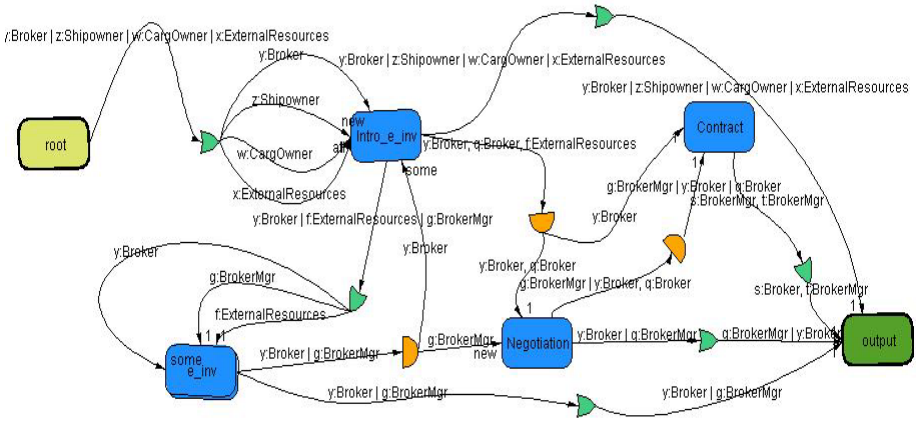


Fig. 5. e-Chartering performative structure (using ISLANDER editor)

According to Figure 1 the performative structure should at least describe the main scenes of the e-Chartering procedure. The extension to these scenes could be implemented (at least in design) by capturing more e-Chartering scenarios. Based on Figure 5 there are four scenes apart from the *root* and the *output* scene [26]. Each scene is created by an institutional agent. The *Shipowner* agent and the *CargOwner* agent together with *ExternalResources* agents and the corresponding *broker* agents are the inputs for the *intro_e_inv* scene which is the *registration scene* for all the participants in each Maritime port (Fig. 6).

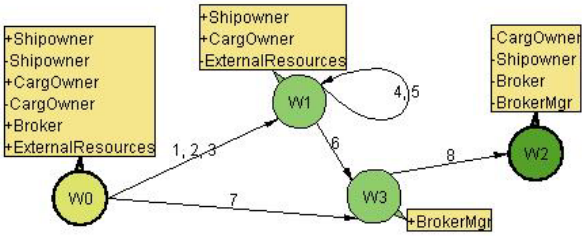


Fig. 6. *intro_inv* scene

The following labels are associated to the arcs in the *intro_inv* scene:

```
1 inform(?s:Shipowner,?b:Broker, position(?position))
2 inform(?c:CargOwner, ?b:Broker, order(?order))
3 inform(?e:ExternalRsc, ?b:Broker, charterinfo(?info))
4 inform(?s:Shipowner,?b:Broker, position(?position))
5 inform(?c:CargOwner, ?b:Broker, order(?order))
6 wait(?timeout_limit)
7 wait(?timeout_limit)
8 inform(?b:Broker, ?bm:BrokerMgr, performed_contracts(?price,?contracts))
```

After the registration and the submission of the position and order info for each agent (Ship owner and Cargo owner), the EI passes into the *e_inv* scene, which is the investigation phase for e-Chartering depicted in Figure 7.

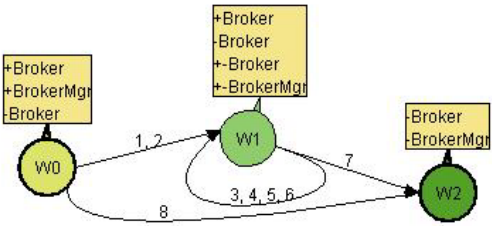


Fig. 7. *e_inv* scene

The following labels are associated to the arcs in the *e_inv* scene:

```
1 inform(?b:Broker,bm:BrokerMgr, want_a_vessel(?order))
2 inform(?b:Broker,bm:BrokerMgr, want_a_cargo(?position))
3 inform(?bm:BrokerMgr, all:Broker, brokering_begins(?position,?sceneId))
4 position(?bm:BrokerMgr, all:Broker, investigate_for_vessel(?order,?sceneId))
5 position(?b:Broker, ?bm:BrokerMgr, investigate_for_cargo(?position, ?sceneId))
6 inform(?bm:BrokerMgr, all:Broker, brokering_begins(?order, ?sceneId))
7 inform(?bm:BrokerMgr, all:Broker,end_investigation())
8 inform(?bm:BrokerMgr, all:Broker,end_investigation())
```

The conflicts to which agents operating in the e-Chartering infrastructure may result to during the search and the investigation stages, are reconciled by the *Broker Manager* agent as follows: The *Broker Manager* controls the collaborative activity within the EI. During collaborative activity, group members must act like a single entity by sharing knowledge, creating common awareness, sharing practices and preferences, and building and maintaining models of their peers [13]. This is particularly true during the investigation phase where the different *broker* agents will search in their domain (port) and then in other ports based on specific search criteria that the Shipowner or the Cargo owner has specified [3]. The result from the search procedure will be a set of different (and maybe contradictory among agents) order or position results. To deal with these cases, the *Broker Manager* possesses policies for reaching a view that the group of all *broker* agents members shall accept [13], acting as a single entity [18][19][20].

The personal agent for the Ship owner or the Cargo owner can offer three alternatives towards the fulfillment of a specific responsibility. In the first mode, the user decides to fulfill the responsibility without the help of his agent, so the job of the *broker* is easier as it doesn't act on behalf of the user. In the second mode, the user delegates the responsibility to the agent. The personal agent must have the appropriate capabilities and knowledge to fulfill the delegated responsibility. In case the delegated responsibility is a collaborative one (like the case where very high level of competition detected at or near destination port), then the agent must have collaboration abilities and must be able to cope with the description of knowledge. In the third mode, the user collaborates with the *broker* agent for the fulfillment of the responsibility. This collaboration with the *broker* agent is independent on whether the responsibility is collaborative, as it concerns the relation between the user and his representative [13][21][22][23].

4.5 Normative Rules

Normative rules are composed by illocutions and meta-language predicates [24]:

$$(s_1, \gamma_1) \Lambda \ldots \Lambda (s_m, \gamma_m) \Lambda \varphi_1 \ldots \varphi_m \Rightarrow \Phi_{n+1} \Lambda \ldots \Lambda \varphi_r \quad (4)$$

where $(s_1, \gamma_1) \Lambda \ldots \Lambda (s_m, \gamma_m)$ are pairs of illocution scenes and illocution schemes, and $\varphi_1 \Lambda \ldots \ldots \Lambda \varphi_r$ are meta-language predicates. The following example shows how normative rules are specified:

```
(shipowner, commit (?x:a, ?y:b, vessel(?size,?port,?type, …)))
⇒ commit (?y:b,?z:c, register(?vessel characteristics))
```

In the e-Chartering case study, when a Ship owner commits a vessel for chartering an order, the vessel is also obliged to commit in a *registration scene* using the vessel characteristics. The deployment of normative rules is motivated by the need of the institution to infer agents' obligations, as well as the consequences of agents' actions across different scenes of agents' participation.

4.6 Policies

The chartering policies are controlled by the *Broker Manager* agent in each EI. This agent will receive contributions from *Broker* agents and compute group acceptances (i.e. chartering solutions) and disseminate these to the group. It is required by each *Broker* agent to inform about changes in its personal contributions the other agents, which follow the same policy [13]. It must be noticed that in each EI (in the same or in different Maritime ports) different policies and the communication requirements they imply could be a serious source of hindering the performance of the MAVCM infrastructure: During the investigation procedure these policies could delay the search and mapping process between Ship owners and Cargo owners, and this is under our future research.

5 Using EIDE for e-Chartering

The Electronic Institutions Integrated Development Environment (EIDE) is a set of tools to support the engineering of Multi Agent Systems (MAS) as EIs [24]. EIDE allows for engineering EIs and their participating software agents. Notably, EIDE moves away from machine oriented views of programming toward organizational inspired concepts that more closely reflect the way in which we may understand distributed applications such as MAS. It supports a top-down engineering approach: Firstly the organization, secondly the individuals. The proposed EIDE for the MAVCM infrastructure development and deployment is composed of ISLANDER [6], SIMDEI [32], aBUILDER [24], AMELI [9] and a Monitoring tool.

5.1 EI Design

The e-Chartering market analysis [3] clarifies in detail all the procedures and constraints that the Ship owner or a Cargo owner should follow. These market rules are the basis for a full specification of EI components. The analysis required for the complete

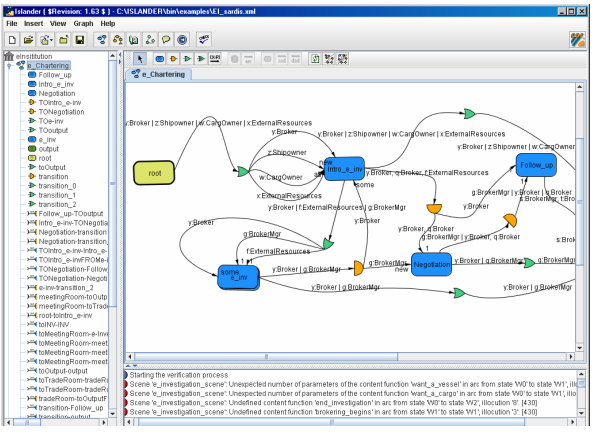


Fig. 8. The ISLANDER editor

specification of the system forces the designer to gain a thorough understanding of the modeled institution before developing it. The analysis phase using a graphical tool as ISLANDER [6] facilitates the work of the institution designer, combining graphical and textual specs of EI components, based on the formalization of EIs [7] (Fig. 8). The conversation protocol in scenes, the relationships among roles in the dialogical framework and the performative structure graph shown in Figure 5, have been specified using the ISLANDER editor. The graphical presentations help the designer to navigate through a structured data presentation of the graphical EI components and provide more insights and deep thoughts of all the elements and sub-elements that belong to the current specification, ordered by category.

5.2 EI Verification

The verification stage for the electronic institution includes both, the *static* aspects of the specifications and the *dynamic* behavior of the EI. The verification of the *static* aspects concerns the structural correctness of specifications based on integrity, liveness, protocol correctness and normative rules' correctness. The *dynamic* verification of the EIs is carried out by simulation procedures under different circumstances and permissions for participating agents.

The tool that supports the static verifications is the *verification message panel* of the ISLANDER tool, shown in Figure 9. Using this tool we were able to perform:

- E-Institution Verifications
- Dialogical framework verifications
- Performative structure verifications
- Scene verifications
- Ontology based verification of messages

The dynamic verification of the e-Chartering EI can be carried out by the SIMDEI simulation tool [8]. SIMDEI supports simulations of EIs with varying populations of agents, enabling *what-if analysis*. For example, SIMDEI can support the simulation of

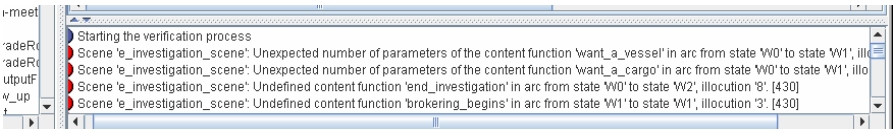


Fig. 9. ISLANDER Verification message panel

the behavior of an e-Chartering procedure between different port brokers for the same Cargo owner order and different Ship owners' positions. The verification process starts with the definition of agents with varying acting capabilities. The dynamic verification procedure is a future task for our MAVCM framework.

5.3 EI Development

The development of agents based on the EI services and duties is the step performed before the EI is deployed and opened to agents playing external roles. Using the aBUILDER tool, the EI designer is able to automatically generate agents' skeletons based on the ISLANDER graphical specifications of agent behaviours. The resulting code is in Java language.

Also the JADE [17] agent platform is an additional option for the development and the implementation of the MAVCM system, due to the large user community, the scope for wide deployment of generated organizations and compliance with FIPA standards [31]. The construction of the agents and the development of the EIs resulting in an operational e-Chartering prototype are within our current goals.

5.4 EI Deployment

The EI operates as a middleware between the participating human agents and the chosen communication layer: It validates agents' actions based on EI rules. The middleware layer is enforced from AMELI. The participating agents in the EI do not interact directly; they have their interactions mediated by the AMELI through a special type of internal agent called *governor* that is attached to agents. The governors support agents by providing the necessary info in order to participate in the institution. AMELI is a general purpose platform where agents consult institutional specifications formed as XML documents generated by ISLANDER. As a result, the implementation impact of introducing institutional changes amounts to the loading of XML-encoded specifications. During an EI execution, the agents composing AMELI maintain and exploit their execution state, along with the institutional rules encoded in the specification, for the validation of actions and assessment of their consequences. AMELI agents are working on handling the institution execution. The EI execution starts with the institution manager. The institution manager activates the initial and the final scenes launching a scene manager for them. The external agents submit their positions/orders for participation in the EI, and when they get an authorization they are connected to the governor and admitted into the initial scene. These agents are allowed to participate in the different scenes executions or to start new scene instances based on the EI specification and the current execution state. The monitoring of the EI execution can be done through the *monitoring tool* which depicts all the

events occurring at run time. It should be noticed that in addition to AMELI there are a number of other frameworks that one may use for building agents organizations, such as PaGoDa [25], MadKit [14], Karma [15], and S-MOISE [16].

The deployment procedure is out of the scope of this paper as our aim was the completion of the design and verification of the e-Chartering procedures using technologies that could ”mirror” the human interactions taking place in real-world Maritime e-Chartering procedures.

6 Conclusions

Designing a MAS system is a very complex and difficult task. This is particularly true for open MAS that are populated by heterogeneous and self-interested agents. Using frameworks like the EIDE [11][12] we managed to effectively design and develop electronic institutions for an open MAS system. EIs define the rules and the constraints for the agent societies, by specifying what is allowed and what is forbidden for them.

The resulting benefits from using the EIDE tools are the shortest design and development cycle and the low cost implementation and execution.

This paper presented and analyzed an EI design, presenting also the development of the suggested solution using EIDE. Using the tools in the design phase, and in specific the ISLANDER editor, we managed to describe all the aspects involved in the e-Chartering scenes, specifying the dialogical framework, scenes and transitions, the performative structure, and the normative rules.

These specifications plentiful supply design details that will be used in our research towards characterizing and checking the development and deployment methodologies, for complex infrastructures as the e-Chartering MAVCM.

References

1. Guinchiglia, F., Mylopoulos, J., Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585. Springer, Heidelberg (2003)
2. Juan, T., Pearce, A., Sterling, L.: ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In: Proceedings of Autonomous Agents and Multi-Agent Systems – AAMAS 2002, pp. 3–10, Bologna, Italy (2002)
3. Sardis, M., Maglogiannis, I.: Agents Methodologies for e-Chartering Market design. In: 4th IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI), Athens, Greece, pp. 175–185 (2007)
4. Dellarocas, C., Klein, M.: Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces. In: Proceedings ACM Conference on Electronic Commerce (EC 1999) (1999)
5. Diller, A.: An introduction to Formal Methods, 1st edn. John Wiley & Sons, Inc, Chichester (1990)
6. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an electronic institutions, editor. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), Bologna, Italy, pp. 1045–1052 (2002)

7. Esteva, M.: Electronic Institutions: from specification to development. IIIA Ph.D. Monographs, vol. 19 (2003)
8. Esteva, M., Rodriguez-Aguilar, J.A., Arcos, J.L., Sierra, C., Garcia, P.: Institutionali-sing open multi-agent systems, A formal approach. Technical report, Artificial Intelligence Research Institute, Spanish Council for Scientific Research, IIIA Research Report 2001-01 (2001) (2000), http://www.iiia.csic.es/Publications/Reports/2000
9. Esteva, M., Rodriguez-Aguilar, J.A., Rosell, B., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Jennings, N., R., Sierra, C., Sonenberg, L., Tambe, M. (eds.). Proc. of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), pp. 236–243 (2004)
10. Noriega, P., Sierra, C.: Towards layered dialogical agents. In: Third International Workshop on Agent Theories, Architectures and Languages, ATAL 1996 (1996)
11. Noriega, P.: Agent-mediated auctions: the fishmarket metaphor. IIIA Ph.D. Monographs, vol. 8 (1997)
12. Rotriguez-Aguilar, J.A., Noriega, P., Sierra, C., Padget, J.: A java-based electronic auction house. In: Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 1997), pp. 207–224 (1997)
13. Partsakoulakis, I., Vouros, G.: Agent-Enhanced Collaborative Activity in Organized Settings. International Journal of Cooperative Information Systems (IJCIS) 15(1), 119–154 (2006)
14. Gutknecht, O., Ferber, J.: The MadKit agent platform architecture. In: Agents Workshop on Infrastructure for Multi-Agent Systems, pp. 48–55 (2000)
15. Pynadath, D.V., Tambe, M.: An automated teamwork infrastructure for heterogeneous software agents and domains. Autonomous Agents and Multi-Agent Systems 7(1-2), 71–100 (2003)
16. Hubner, J.F., Sichman, J.S., Boissier, O.: S-MOISE: A middleware for developing Organized Multi Agent Systems. In: Proc. of the AAMAS 2005 workshop: From Organizations to Organization Oriented Programming (OOOP) (2005)
17. JADE: Java Agent Development Framework, http://jade.tilab.com/
18. Dignum, V., Meyer, J.J., Weigand, H., Dignum, F.: An organization-oriented model for agent societies. In: Proceedings of RASTA workshop (AAMAS) (2002)
19. Odell, J., Van Dyke Parunak, H., Fleischer, M.: The Role of Roles in Designing Effective Agent Organizations. In: Garcia, A.F., de Lucena, C.J.P., Zambonelli, F., Omicini, A., Castro, J. (eds.) Software Engineering for Large-Scale Multi-Agent Systems. LNCS, vol. 2603. Springer, Heidelberg (2003)
20. Vazquez-Salceda, J., Dignum, V., Dignum, F.: Organizing Multiagent Systems. Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
21. Grosz, B.J., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
22. Khalil-Ibrahim, I., Kotsis, G., Kronsteiner, R.: Substitution Rules for the Verification of Norm-Compliance in Electronic Institutions. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2004), 1524-4547/04 (2004)
23. Purvis, M., Savarimuthu, S., de Oliveira, M., Purvis, M.: Mechanisms for Cooperative behaviour in Agent Institutions. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006), 0-7695-2748-5/06 (2006)
24. Arcos, J.L., Esteva, M., Noriega, P., Rodriquez-Aguilar, J.A., Sierra, C.: Engineering open environments with electronic institutions. In: Engineering Applications of Artificial Intelligence 18, pp. 191–204. Elsevier, Amsterdam (2005)

25. Partsakoulakis, I.: Team-oriented behaviour in dynamic agent organizations. PhD Dissertation in Greek (2007)
26. Vazquez, J., Dignum, F.: Modeling electronic organizations. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 584–593. Springer, Berlin (2003)
27. Rodríguez-Aguilar, J.A.: On the Design and Construction of Agent-mediated Electronic Institutions. IIIA Phd Monographs, Universitat Autonoma de Barcelona, Vol. 14 (2001)
28. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Transactions on Software Engineering and Methodology 12(3), 317–370 (2003)
29. Sierra, C., Jennings, N.R., Noriega, P., Parsons, S.: A Framework for Argumentation-based Negotiation. In: Rao, A., Singh, M.P., Wooldridge, M.J. (eds.) ATAL 1997. LNCS, vol. 1365, pp. 177–192. Springer, Heidelberg (1998)
30. Rahwan, I., Sonenberg, L., McBurney, P.: Bargaining and Argument-Based Negotiation: Some Preliminary Comparisons. In: Rahwan, I., Moraïtis, P., Reed, C. (eds.) ArgMAS 2004. LNCS (LNAI), vol. 3366, pp. 176–191. Springer, Heidelberg (2005)
31. FIPA-The Foundation for Intelligent Physical Agents (March 2003), http://www.fipa.org
32. REPAST, http://repast.sourceforge.net
33. FIPA-Agent UML (2005), http://www.auml.org
34. Bauer, B., et al.: Agent UML: A Formalism for Specifying Multiagent Interaction. In: Ciancarini, P., Wooldridge, M. (eds.) Agent-Oriented Software Engineering, pp. 91–103. Springer, Berlin (2001)
35. The Digital Ship, http://www.thedigitalship.com/DSmagazine/digital%20ship%20archive/digital%20ship%20nov/echartering.htm#_Toc528400402
36. Batrinca, G.: E-chartering web based platforms between success and failure. In: TRANSNAV 2007, Gdynia, Polland (2007)
37. Maritime e-Commerce Association, "Standards of E-chartering", http://www.meka.org.uk/standards.asp?standardsID=2

Multi-agent Simulation to Implementation: A Practical Engineering Methodology for Designing Space Flight Operations

William J. Clancey[1], Maarten Sierhuis[2], Chin Seah[3], Chris Buckley[4], Fisher Reynolds[4], Tim Hall[5], and Mike Scott[3]

[1] NASA Ames Research Center, Intelligent Systems Division, Moffett Field, CA 94035, USA
[2] RIACS, NASA Ames Research Center
[3] QSS, NASA Ames Research Center
[4] USA, NASA Johnson Space Center
[5] NASA Johnson Space Center
{William.J.Cancey,Maarten.Sierhuis-1,Christopher.B.Buckley, f.f.reynolds,Timothy.A.Hall}@NASA.Gov, CSeah@mail.arc.nasa.gov, mscott@ptolemy.arc.nasa.gov

Abstract. OCAMS is a practical engineering application of multi-agent systems technology, involving redesign of the tools and practices in a complex, distributed system. OCAMS is designed to assist flight controllers in managing interactions with the file system onboard the International Space Station. The "simulation to implementation" development methodology combines ethnography, participatory design, multiagent simulation, and agent-based systems integration. We describe the model of existing operations and how it was converted into a future operations simulation that embeds a multiagent tool that automates part of the work. This hybrid simulation flexibly combines actual and simulated systems (e.g., mail) and objects (e.g., files) with simulated people, and is validated with actual data. A middleware infrastructure for agent societies is thus demonstrated in which agents are used to link arbitrary hardware and software systems to distributed teams of people on earth and in space—the first step in developing an interplanetary multiagent system.

Keywords: Work Systems Design, Work Practice Simulation, Decision Support System, Multi-Agent System, Agent-based Systems Integration, Space Flight Operations.

1 Introduction

OCAMS (Orbital Communications Adapter Mirroring System) is a practical engineering application of multi-agent systems technology, using the Brahms modeling and simulation tool [1-6], involving redesign of the tools and practices in a complex, distributed system. The purpose of the project was to automate some of the tasks involved in mission operations at the Mission Control Center (MCC) at NASA Johnson Space Center (JSC), supporting the International Space Station (ISS).

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 108–123, 2008.

The combination of people and systems involved in JSC mission operations support is complex and distributed [7]. When long-term complex programs like ISS evolve, new systems and processes are introduced that interact with legacy systems. This creates a growing distributed systems environment that can be taxing on the flight controllers and introduce more risk of human error. The OCAMS solution bridges these complex distributed systems and automates processes that are repetitive and time consuming. This simplification helps improve operator productivity and safety and reliability by reducing the chances of human error.

The OCAMS tool is complex because it is embedded in the infrastructure of geographically and temporally distributed people and systems for which it facilitates communication:

1) **People and Organizations:** Flight controllers, "backroom" support teams, and customers (planners, human factors specialists, etc.) located in different rooms at MCC and at other NASA centers in other states; and the crew onboard ISS.
2) **Computer Systems:** File servers, PCs communicating with the ISS, support PC, the PC that mirrors the ISS file directories (MirrorLAN), and PC laptops onboard ISS.
3) **Communication Media:** Voice communications system ("voice loop") at MCC, telephone, email, "flight notes," "change requests," and log documents.
4) **Space Communication Network:** Communication between ground and ISS using the TDRS satellite system is short-term, periodic and irregular (from human perspective).
5) **Out of this world geographic distribution:**
   a) Multiple NASA centers
   b) In Houston: Highly secure Mission Control Center with flight control rooms and support backrooms; offices in different buildings at JSC
   c) International partners' (Russia, Europe, Japan) control rooms and offices
   d) ISS orbiting earth about every 90 minutes.
6) **Regulations relating to safety and control have over time produced disconnected, legacy systems:**
   a) no ISS link to earth's Internet
   b) no cell phones in MCC
   c) no network connection between MirrorLAN and MCC file servers
   d) multiple versions of operating systems and file generation and handling programs at different NASA centers and onboard ISS.
7) **Work Practices and Protocols:**
   a) Diversity of methods for delivering files, notifying support personnel (called "officers") of work to be done (see Communication Media), and notifying customers (usually "flight controllers") of completed tasks and problems
   b) Continuous 24-7 coverage in three shifts (called "orbits")
   c) Shift handovers relying on detailed logs created manually documenting the work done on the previous shift, anticipated work, and ongoing issues.

The OCAMS tool was designed in a collaboration between two NASA centers, JSC (an operations center) and Ames Research Center (ARC). The objectives included: 1) Developing new mission operations design and automation capabilities that would reduce the need for ISS ground support on a 24-7 schedule, and 2) Shifting MCC's concept of operations from controlling systems directly onboard the ISS to supporting astronauts living and working in space.

The project approach was to automate tasks to improve operator productivity, increase accuracy of the process, and eventually enable consolidation of this position into other console disciplines. A year was allowed to demonstrate a new methodology and automation capability, in which we would use a multiagent system to simulate and implement an automation tool. This paper describes the methodology and presents the results of the project in the first half year, including partial implementation of a prototype tool within a simulation of the new work system (called a "future operations simulation").

More broadly, in terms of engineering agent societies, this project illustrates the following themes:

1) **Highly-interdisciplinary methodology for the engineering of complex distributed applications**
   a) Ethnography
   b) Mission Operations (flight controllers & protocols)
   c) IT Administration: tools and constraints (OCA–Orbital Communications Adapter wireless card, servers, FTP, email, multiple networks, security, file types, mirroring, GUI, agents)
   d) Brahms: Work Practice Simulation
   e) MA: Agent-based Systems Integration
   f) Java platform
2) **Analysis, Design, Development & Verification of Agent System**
   The simulation to implementation methodology enables dealing with complexity by using a simulation to design and largely implement a tool that is integrated with a simulation of the work setting and practices:
   a) The future tool is embedded in the Future Operations Simulation
      i) Simulated people (Brahms agents)
      ii) Actual software agents (e.g., personal agents)
      iii) Actual external systems (e.g., email, FTP, file system, office tools)
      iv) External system APIs used by Brahms Communication (COM) Agents
   b) The work system design and tool is tested with actual data
      i) Simulation is driven by the same inputs used by the future tool
      ii) Develop using part of the data set (e.g., a month's input)
      iii) Continue to test during the implementation phase by using new data as it becomes available.
3) **Middleware infrastructures for agent societies: Use of Brahms agents and external system APIs to link arbitrary hardware/software to teams of people**
   Brahms provides a promising candidate for answering the question: How will we build practical complex agent systems on a variety of platforms using arbitrary external software and hardware devices?

This paper describes the OCAMS project's origin and scope, the Current Operations Model and simulation output and the consequential design and partial implementation of a multiagent tool (OCAMS) that automates operations in what we call the Future Operations Simulation. Conclusions review how the development of OCAMS provides methods and insights for engineering agent societies.

2 Simulation to Implementation Approach

Our methodology makes multiagent simulation of work practices an integral part of creating agent software, an approach we call "simulation to implementation" (Fig. 1). The approach starts with the creation of a *Current Work Practice Simulation Model* (CSM), using the Brahms language. The purpose of this simulation is to help frame an organization's problems and prioritize relationships and trade-offs. For example, how will the functionality of NASA's new spacecraft vehicle—the CEV—impact mission operations, and how will the vehicle to ground split in functionality impact communications and in turn performance? Framing the problem to be addressed, metrics and scenarios are developed to create a work practice model in Brahms. Simulating the model in Brahms will generate simulation data that can be used to interpret the outcome and validate the answers to the framing questions. One important additional aspect is the use of such a simulation to generate new ideas for formulating the problem, and ultimately identifying solutions to the problems that can be addressed in the next design phase. Methods used in this phase are work practice observation (including videotaping and still photography), collaborative modeling with the workers from the organization, and interview techniques [8].

In the *Participatory Design of Future Work System* phase we work closely with the workers from the organization to design a solution to the problems identified in the current work practice simulation [9]. In this phase we generate user-driven requirements and turn these requirements into a functional and technical design of a multiagent workflow tool. Following a principle of participatory design—transforming current practices rather than believing one can start from scratch—leads to the *Future Work System Simulation Model* (FSM) phase, in which the CSM model is adapted to include the proposed tool(s). The data, metrics. and scenarios from Phase 1 are used to drive the future work simulation, allowing comparison of the CSM with the FSM models and validating the improvements of the new design. Because the future tool is embodied in the FSM model, from the workers' perspective it is actually a prototype tool that runs in an automatic mode driven by historical data, simulating human actions. By providing interactive control of the simulation in a prototype GUI, the tool's operation can be demonstrated and its automatic features inspected and hence refined.

In the *Work System Implementation* phase we transform the Brahms FSM model into a distributed real-time multi-agent system (MAS). The Brahms simulation engine in runtime mode will shift the discrete event simulation from being driven by an internal clock to being coupled to the time and events in the real world, thus transforming simulated agents into real-time software agents. Brahms can both simulate or execute its agent models over the internet, enabling a seamless transformation from an agent-based simulation environment to a distributed multi-agent system environment.

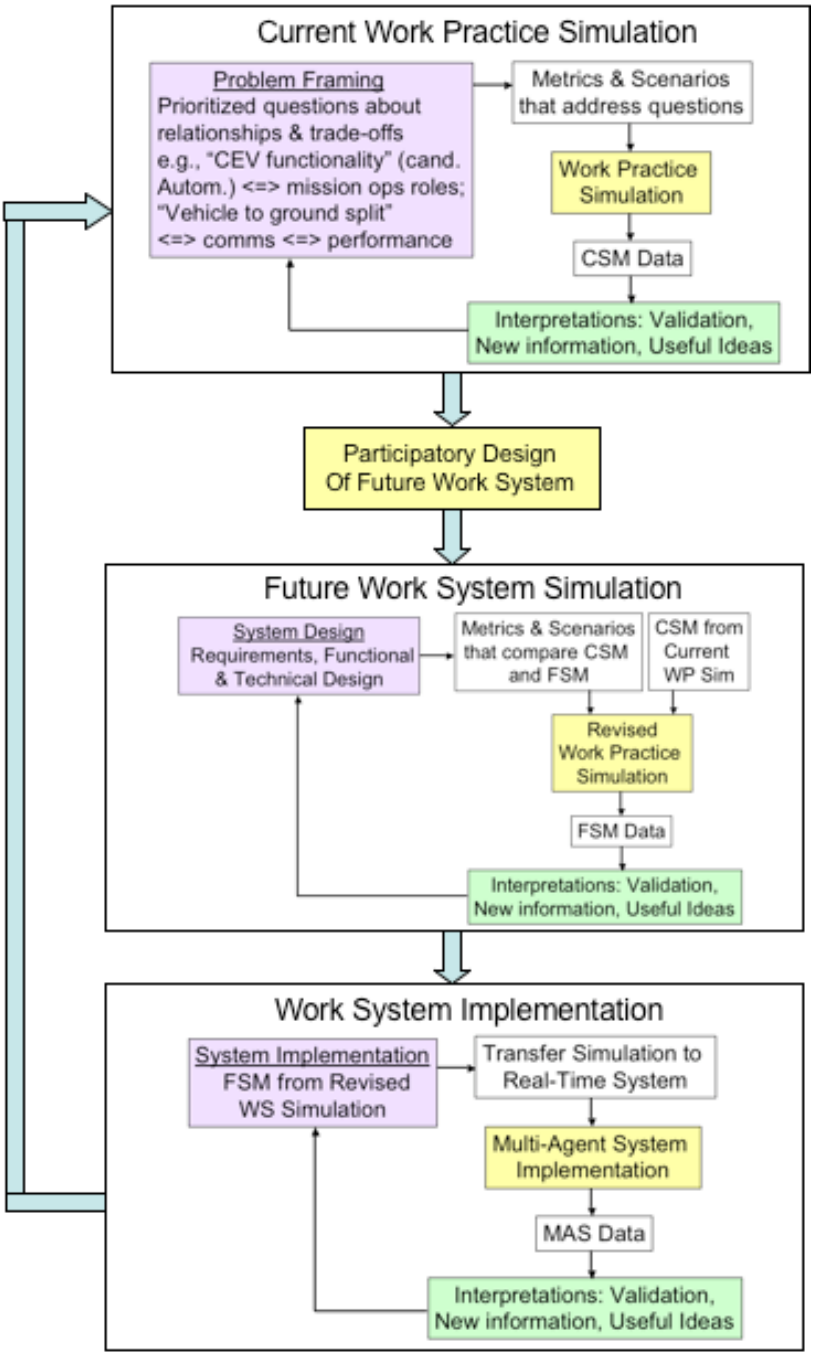


Fig. 1. Simulation to Implementation Approach

3 Project Origin and Scope

One purpose of this project was to demonstrate the use of an agent-based simulation-to-implementation methodology in Mission Control at NASA's Johnson Space Center. Program management chose the Orbital Communications Adapter (OCA) backroom group, which provides file transfer support to the ground team and astronauts onboard the ISS.

Applying the simulation-to-implementation methodology to the OCA setting involved the following activities:

1) Observation of OCA operations and interviews with OCA officers
2) Creation of a baseline simulation of current operations using Brahms
3) Collaborative redesign of the work system (documented in a functional specification)
4) Creation of a future operations simulation that embeds an agent-based workflow automation tool, implementing the functional specification (documented in a technical design)
5) Validation of the tool and revised work processes by driving the simulation with actual data
6) Integration of the agent-based workflow tool in the MOD work environment.

In this methodology, multiagent simulations serve multiple roles for understanding, communication, formalization, specification, validation, and implementation.

The purpose of the OCA current operations model was to create a baseline understanding and formal description of an aspect of the OCA work process that could be redesigned. Early observations of OCA operations and discussions with OCA officers indicated that mirroring of ISS files was a good candidate for improvement. Given time constraints and modelers available, our strategy in developing the current operations model was to understand and simulate enough of the system to provide confidence that we could develop a functional specification for automating the mirroring process. Consequently, the simulation does not attempt to capture any of the timings or activities of the OCA officer in any detail, except for the mirroring activity. The simulation showed that the OCA officers spend about 6% of their work time on the mirroring activity.

The development of a current operations simulation has also served as tool for management to understand the Brahms agent-based architecture and to grasp how a future operations simulation could be converted into a workflow tool. The future operations simulation is described at the end of this paper. By virtue of formalizing the future design with a prototype GUI, it serves the additional role of a management decision support tool for redesigning mission operations.

4 Model of the OCA Current Operations Work System

This section provides an overview of the OCA current operations model, which represents the typical actions of OCA officers during a shift. The activity model describes what the OCA officer does during the shift; only the mirroring activity is modeled in any detail in this current operations simulation. The main components of the model

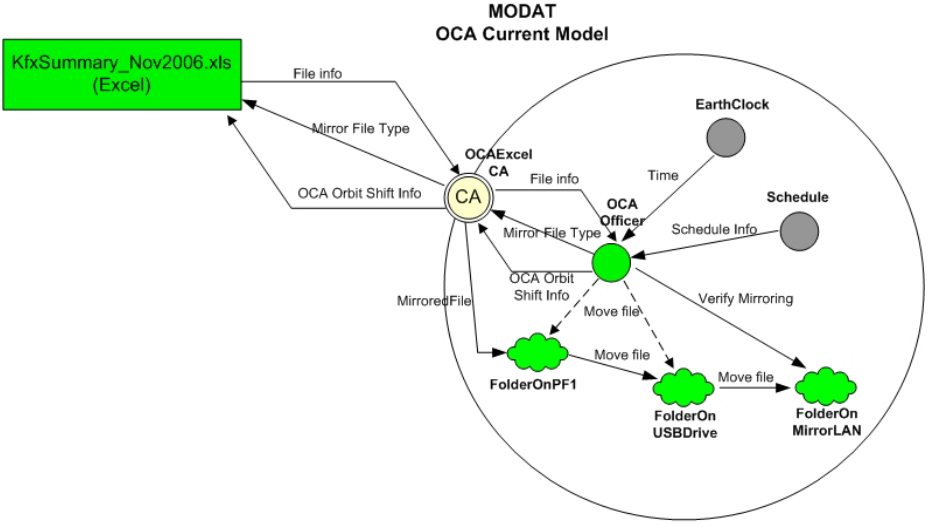


Fig. 2. Agents and Objects in the OCA Current Operations Model

Table 1. Example of Kfx Summary Log data that drives the current operations simulation

Up/Down	Downlink	Uplink
GMT	300/23:59:00	301/00:00:05
Bytes	1,364	40,539,150
FileName	d:\oca-down\Updates.log	U:\COSS\ILRT\Ref CD42\TrainingManuals\English\01(0)T0008E.pdf
Extension	log	Pdf
Client	Plan D	Plan B
Year	2006	2006

are: OCA Officers, the OCA computers and drives, Building 30S (the MCC) work areas, computer files and folders, and work schedules. Fig. 2 shows the agents and main flow of data and commands between the agents and objects in the OCA current operations model. For simplicity, the folders are represented as geographic areas (shown as clouds).

To drive the simulation, we used a spreadsheet provided by an OCA officer, KfxSummary_Nov2006.xls, which had been derived using macros from a log created automatically by the ISS uplink/downlink software of the November 2006 file transfers. Table 1 shows an entry from the spreadsheet.

Referring to Fig. 2 notice that a special agent, called the OCA Excel Com Agent (Excel CA, hereafter ECA) provides information about what files were transferred during a particular shift (as shown in Table 1). The simulated OCA Officer agent determines whether a given file needs to be mirrored based on its type. The ECA simulates the file being placed in the location FolderOnPF1. The OCA Officer agent then operates on the file using PF1, the USB Drive, and the MirrorLAN.

1. The OCA Officer agent sends the ECA its shift information at the beginning of its shift just after handover:
    - GMT Date, e.g. 305 is Nov 1st
    - GMT Start/Stop Hour and Minute
2. ECA sends file information back to OCA agent:
    - File Extension, e.g. pdf, xml, zip
    - File Name without Path, e.g. nfhWednesday.pdf
    - File Path, e.g. /BHPG/Crew/News/
    - File Direction (Uplink or Downlink, where "up" means to the ISS)
3. OCA Officer agent applies thoughtframes that use file type information and sends back to ECA: Decision to mirror or not (true or false) and File Type symbol, e.g. OSTPV_Snapshot_File_Type.
4. If the file is being mirrored, the ECA then puts the file in the location FolderOnPF1, and informs the OCA Officer agent of the location and File Type symbol.

This part of the simulation is not a model of work practice, but rather a method of driving the simulation to use actual file transfer data. The effect is that the simulated OCA Officer agent will mirror the same files during a given simulated shift that were mirrored in the corresponding actual shift, by virtue of processing the files listed for that time period in the Kfx Summary Log file.

The simulation constitutes a model of work practice (i.e., has fidelity) by virtue of including the following:

- Data about file transfers that can be derived from the Kfx logs, including file names, paths, sizes, and transfer direction.
- Relationship between file path/name and type of file (Table 1), e.g., ACKBAR files, BEV updates files, DOUG files.
- OCA officer activities of transferring files from PF1 to USB Drive to MirrorLAN, in which the duration of these activities is estimated by the actual byte size of the files being transferred at any time.
- OCA officer activities of monitoring file processing by services running on the MirrorLAN, in which the duration of these activities were estimated by OCA officers, based on file type.

Consequently, statistics can be generated from the model regarding how much time the OCA agent spends mirroring files. Furthermore, by virtue of recognizing file types, procedures for handling different types (e.g., providing notification) can be modeled more easily in the Future Operations Simulation.

The Brahms Current Operations model completely describes an OCA shift. However, only the shift handover, file transfer, mail synchronization, and mirroring operations are modeled in any detail. The behaviors of people (modeled as Brahms agents), systems (modeled as Brahms objects), and software agents (modeled as Brahms agents) are represented as Brahms workframes and thoughtframes.

Here is an example of one of the actions performed by the OCA Officer agent beginning the shift (ReadOCAHandoverLog is abbreviated ROHL):

```
workframe Read_OCA_Handover_Log {
     variables:
         forone(int) maxTime; minTime; actPriority;
     when(
         knownval(current.currentIndividualAct = ROHL) and
         knownval(ROHL.isDone = false) and
         knownval(maxTime = ROHL.maxDuration) and
         knownval(minTime = ROHL.minDuration) and
         knownval(actPriority = ROHL.activityPriority))
     do {
     moveToIndividualActivityLocation();
     ROHL(actPriority, minTime, maxTime,
         Statistics_Understanding);
     conclude((ROHL.isDone = true), fc:0);}
} // workframe Read_OCA_Handover_Log
```

In the conditional or "when" part of the workframe, the durations are read from the activity schedule object ReadOCAHandoverLog. In the action or "do" part of the workframe the agent does the following: 1) moves to an appropriate location, 2) reads the log (a primitive activity performed for the specified time), and 3) concludes that the activity of reading the log has been done. (Such propositions become part of the individual agent's model of the world and are called *beliefs*, contrasted with the Brahms global model of the world, consisting of *facts*, which are only accessible to agents via uncertain *observables* that occur during activities [1][5].)

Reading the log is defined as a primitive activity as follows:

```
primitive_activity ReadOCAHandoverLog(int priorityNum,
int minDuration, int maxDuration, Statistics statObj) {
     display: "Read OCA Handover Log";
     priority: priorityNum; random: true;
     min_duration: minDuration; max_duration: maxDuration;
     resources: statObj; }
```

The resources property indicates that statistical information about this activity should be logged by the simulation engine. See Section 4.2 below on Statistical Charts. The min_duration, max_duration and random facets of the activity definition specify a random duration at runtime.

5 OCA Current Operations Simulation Output

The Current Operations model was run for 31 simulated days, corresponding to the OCA officers' shifts in November 2006. The simulation result can be verified and validated using two methods, the AgentViewer and statistical charts.

5.1 AgentViewer

A Brahms simulation produces a history file in the form of a database that can be diagramed and studied in the AgentViewer. This allows us to understand the behavior of agents and objects during the simulation. Fig. 3 shows agent behaviors chronologically as activities; Workframes (darker shade) shown with "wf", Primitive

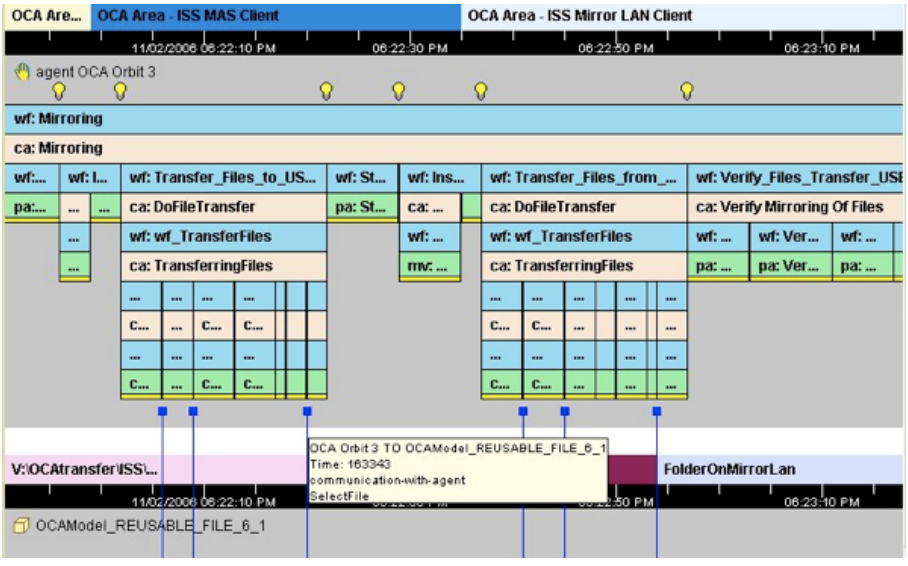


Fig. 3. Simulated OCA agent's actions during Orbit 3 (starting 5:50 PM CST; each white mark is a clock tick = 5 minutes). Behaviors are modeled as workframes with composite activities (CA) that invoke workframes, ending in primitive activities, such as communications (shown as vertical lines connecting to file objects that are not visible here).

Actions (at the bottom) shown with "pa:", Composite Activities (light shade) shown with "ca:", Communications (vertical lines), and Thoughtframe conclusions (light bulbs). Locations appear in the bar above the black timeline for each agent or object.

The agent is transferring files to a USB drive (sitting at ISS MAS client area) and then mirroring the files to the MirrorLAN (at the MirrorLAN area). Labels that won't fit are shown as three dots (…). One communication has been selected, causing details to pop up. Light bulbs may also be selected for details about the belief concluded by a Thoughtframe.

5.2 Statistical Charts

A Brahms simulation can be "instrumented" by defining a resources property for primitive activities (i.e. actions). For example, the communication action SelectFile has the resources property Statistics_Transferring_Files, an object. The primitive activity DraggingFile has the resources property "file," which is the object being manipulated. When the Brahms executive (simulation engine) encounters a resources property, it logs data about the agents, objects, and durations during which that resource was worked on. The current operations model is annotated by 20 statistics categories. These categories represent general work "chunks" classifying the different activities of the OCA officer for which we want the simulation to generate decision-metrics. Fig. 4 illustrates the kinds of charts that can be generated from the resulting statistics.

Analysis of such charts revealed that the OCA Officer spends most of the shift logging and verifying file transfers. Our design has therefore focused on automating the

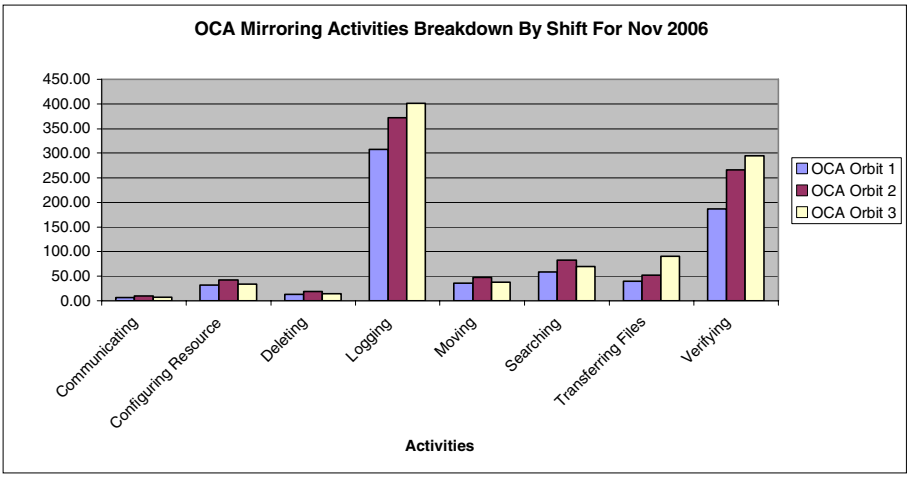


Fig. 4. Example of simulation results for mirroring subactivities. Note: Not definitive data; these charts represent work in progress. They are not necessarily accurate and do not reflect later changes made to the model.

mirroring activity in such a way that these subactivities in the future do not have to be performed by the OCA officer. Besides eliminating a manual error-prone process, automating the mirror activity will result in the OCA officer saving time and potentially enable the position to be given less tedious responsibilities.

More specific charts compare shifts according to the percent of total files transferred to the percentage of files in a given shift (called an orbit) that are mirrored. For example, Orbit 2 (the daytime shift) processed 36% of the files transferred between ground and ISS during November 2006. Of the files that Orbit 2 transferred about 31% were mirrored. Thus, on average about a third of the files processed by an OCA officer are mirrored. This represents a significant workload (about 2500 files manually manipulated) and further justifies automation.

In developing the future operations simulation, which will change the work design by including a workflow tool for mirroring, we could choose to model additional activities, such as communications with flight controllers, and instrument these events to gather statistics from the simulation. These statistics can then be compared to observations we make of OCA operations, leading us to refine the model or gain confidence in the simulation's predictions. In particular, we could use the simulation to predict that the OCA officer could take on other responsibilities, and include these in another future operations simulation. In this way, we proceed through observation, collaborative design, simulation, and redesign to incrementally improve how the work is done, gaining efficiency and reliability.

6 Creating a Future Operations Simulation

The steps in creating an OCA future operations simulation include: 1) Creating a functional design of the revised OCA work system, including automation of mirroring,

2) Revising the current operations model to more accurately represent the aspects of work pertinent to mirroring automation, 3) Implementing the functional design as Brahms agents, revised OCA Officer agent activities, and a simulated GUI for human-agent interactions, 4) Validating the simulation using existing Kfx Summary logs and carrying out "what if" simulations that introduce problems (e.g., unavailable systems or errors during mirroring).

At the time of this writing, the project is in step 3, implementing the revised work system design as a Future Operations Simulation. The simulation will include approximately 80% of the OCAMS tool. Using the tool, the work process is modified, such that the OCA Officer will perform the following operations:

1. Select files to mirror (reviewing Mirroring Decision Agent's selections)
2. Submit session (a batch) of files to mirror
3. Review and verify results; delete session of mirrored files
4. Handle files not mirrored by OCAMS manually
5. Handle files with MirrorLAN errors identified by OCAMS
6. Review mirrored files automatically logged in handover document
7. Notify flight controllers mirroring completed.

Fig. 5 shows the first version of the future operations simulation. It shows the flow of shift information between the OCA Officer, the Mirroring Decision Agent, and the Monitoring Agent. Files to be mirrored are transmitted by FTP to a staging area on a separate computer where the Monitoring Agent individually moves files to the MirrorLAN (via a drive mapping) and inspects the outcome of batch file execution.

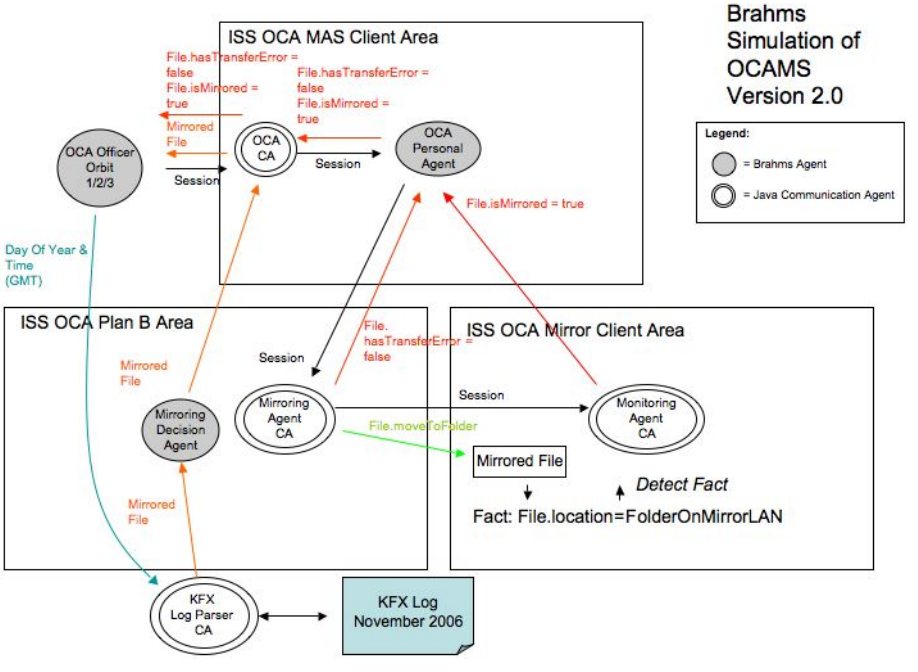


Fig. 5. Future Operations Model, derived from Current Operations Model (Fig. 2)

Through collaboration among the ARC project team, JSC OCA officers, and management, the Future Operations Simulation will be modified to adjust the design of the OCA work system (e.g., as may be required for implementation in the MCC). After it is agreed that the design is complete and validated through simulations, a runtime distributed Brahms agent system can be extracted from the Future Operations Simulation and packaged as the OCAMS tool, to run in the Brahms virtual machine on computers and the network in the OCA backroom area of the MCC. A certification process, to be defined, might include constructing a mockup of this network and operations.

Continuing ethnographic observations of OCA operations will be a key part of our work while simulating the future operations to verify the simulations and to understand how mirroring operations interact with other aspects of work practice, such as notification to other flight controllers. Furthermore, we know from observation that mirroring is a useful training ground for new OCA officers. Therefore, we want to implement the system in such a way that manual operations on the MirrorLAN are still possible and that automated processes adapt accordingly. Similarly, after implementation of the OCAMS tool, an important new phase of observation will begin to understand changes to the work practice, emergent uses of the tool, and ways to improve it.

7 Related Work

In this section we compare and contrast the Brahms simulation framework to Workflow Management Systems and Agent-Based Modeling and Simulation (ABMS) to show the modeling requirements of the simulation to implementation approach.

7.1 Workflow Management Systems

Workflow Management Systems (WfMS) have evolved from business management, business process reengineering, business process modeling and simulation, and to a lesser extent artificial intelligence. OCAMS is a WfMS, where the automated process is not a business, but mission operations.

Recently, workflow modeling languages and tools have been developed as industry standards. For example, Business Process Execution Language (BPEL) is an XML-based language with structured, executable programming concepts that can be integrated with web services. The most common language in academia is the Petri-Net language [10, 11], which uses complex state transition diagrams to model programs and parallel processes such as concurrent tasks. A multi-agent system can be modeled as parallel Petri-Nets.

Workflow models are based on a functional flow-based abstraction of the work, modeling defined tasks and operations such as those formalized in business procedures. In contrast, Brahms' activity-based approach [2] enables modeling the complexity of activities, communication practices, relationships, and circumstantial details of coordination and workarounds, together constituting *work practice*, by which functions are accomplished [12] [9]. Although I-X [13] also uses the concept of an activity, its framework uses a task-planning approach. A Brahms activity is a

broader concept, including more than goal-directed problem solving, such as resting by informally conversing with co-workers [2].

In systems with run-time capabilities, the work process model is or can be automatically transformed into an executable language. For example, BPNM, IDEF3, colored Petri-Net and YAWL languages are imperative programming languages. In contrast, Brahms is an agent-oriented Belief-Desire-Intention (BDI) language which represents processes as an organization of agents with individual beliefs, coordinating group and agent-specific activities represented as situation-action rules [14]. Rather than only expressing functional transformations, Brahms enables representing roles, points of view, habits, temporal rhythms of behavior, contextual factors, communication media, tools, conversations, etc. This level of specificity enables a Brahms agent that automates work to *fit* into the practices of the people who must interact with it, an understanding encouraged by and enabling the embodiment of participatory design within a simulation-to-implementation engineering approach.

7.2 Agent-Based Modeling and Simulation

Agent-based Modeling and Simulation (ABMS) is a term mostly used by researchers in complex adaptive systems to model systems of relatively simple agents that derive their emergent behavior from the system as a whole, instead of from complexity within the agents themselves. Tools for ABMS such as Swarm [15] and Repast [16] are not based on any particular human behavior theory and are not BDI languages; agent methods are driven by a global scheduler. In contrast, Brahms models cognitive agents; their internal state (possible and incomplete activities, plus beliefs, which can represent plans and goals) combined with a complex modeled environment determines the agents' next behaviors. Thus the Brahms language is both a BDI agent language and an ABMS language, which is important for example in representing decision making in mirroring files and handling errors.

In the category of BDI languages [17], Brahms is distinguished from systems such as Jason and AgentsSpeak by its use of a subsumption-based architecture [18] for representing an agent's conceptualization of activities as parallel-hierarchical processes [1, 2]. This allows modeling how activities are like identities that blend and contextually change what is perceived, how communications are interpreted, and how tasks are prioritized. See [14] for additional comparisons to agent-oriented languages.

8 Conclusions

The OCAMS agent system is designed to automate workflow deterministically, under OCA officer control to develop trust, enable customization, manage problems/ shortcomings, and retain a manual approach for use in training. A key aspect of this practical engineering project is the highly interdisciplinary team that partners operations personnel with researchers and combines specialized knowledge from computer science, anthropology, spaceflight operations, and work systems design.

The use of Brahms demonstrates how agents can be used in a "simulation to implementation" methodology by which a model of current operations is converted into a future operations model that incorporates both essential aspects of an agent-based tool and a simulation of how the tool interacts with people and other systems. This

hybrid simulation enables flexible, incremental development of an implementation, such that actual systems (e.g., email, FTP, files) replace simulated systems and objects. The simulations are driven by logs of the actual work performed in the past, and the future operations simulation operates upon the actual files manipulated by the OCA officers. By running the simulation subsequently with data from other months, we can validate the generality of the mirroring rules and special handling designed into the tool.

OCAMS is one of the first steps in developing an interplanetary multiagent system that integrates people on earth and astronauts with a diversity of hardware and software systems. The combination of agent-based simulation and systems integration enables great efficiency in designing, validating, and deploying practical tools.

Acknowledgments. Brahms was originally developed 1992-1997 as a joint project between NYNEX Science & Technology and The Institute for Research on Learning [1, 5], and reengineered in Java at ARC from 1998-2001. The runtime form of Brahms was developed in the Mobile Agents project [10]. We are grateful to Tom Diegelman at NASA JSC for promoting applications of Brahms to mission operations design and securing seed funding for this project in 2006 [11]; Brian Anderson, Dennis Webb, and Ernie Smith also provided essential support. Several other OCA officers not listed as co-authors reviewed and commented on the OCAMS specifications, including Skip Moore and Karen Wells. This project has been supported in part by funding from NASA's Constellation Program.

References

1. Clancey, W.J., Sachs, P., Sierhuis, M., van Hoof, R.: Brahms: Simulating Practice for Work Systems Design. International Journal on Human-Computer Studies 49, 831–865 (1998)
2. Clancey, W.J.: Simulating Activities: Relating Motives, Deliberation, and Attentive Coordination. Cognitive Systems Research 3(3), 471–499 (2002)
3. van Hoof, R., Sierhuis, M.: Brahms Language Reference (2000), `http://www.agentisolutions.com/documentation/language/ls_title.htm`
4. Seah, C., Sierhuis, M., Clancey, W.J.: Multi-agent Modeling and Simulation Approach for Design and Analysis of MER Mission Operations. SIMCHI: Human-computer interface advances for modeling and simulation, January 2005, pp. 73-78 (2005)
5. Sierhuis, M.: Modeling and Simulating Work Practice; Brahms: A Multiagent Modeling and Simulation Language for Work System Analysis and Design. In: Dissertation in Social Science Informatics (SWI), The Netherlands. SIKS Dissertation 10, University of Amsterdam, Amsterdam (2001)
6. Sierhuis, M., Clancey, W.J., Seah, C., Trimble, J.P., Sims, M.H.: Modeling and Simulation for Mission Operations Work System Design. Journal of Management Information Systems 19(4), 85–129 (2003)
7. Sierhuis, M., Clancey, W.J., Seah, C., Acquisti, A., Bushnell, D., Damer, B., Dorighi, N., Edwards, L., Faithorn, L., Flueckiger, L., van Hoof, R., Lees, D., Nandkumar, A., Neukom, C., Scott, M., Sims, M., Wales, R., Wang, S.-Y., Wood, J., Zhang, B.: Agent-based Mission Modeling and Simulation. In: Agent Directed Simulation 2006; part of the 2006 Spring Simulation Multiconference, Huntsville, AL (2006)

8. Blomberg, J., Giacomi, J., Mosher, A., Swenton-Wall, P.: Ethnographic Field Methods and Their Relation to Design. In: Schuller, A.N.D. (ed.) Participatory Design: Principles and Practices, pp. 123–155. Lawrence Erlbaum Associates, Hillsdale (1993)
9. Greenbaum, J., Kyng, M. (eds.): Design at Work: Cooperative design of computer systems, Hillsdale. Lawrence Erlbaum, NJ (1991)
10. van der Aalst, W.M.P.: Putting Petri Nets to Work in the Workflow Arena. In: van der Aalst, J.M.C.W., Kordon, F., Kotsis, G., Moldt, D. (eds.) Petri Net Approaches for Modelling and Validation (pp, pp. 125–143. Lincom Europa, Munich (2003)
11. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 245–275 (2005)
12. Sachs, P.: Transforming Work: Collaboration, Learning, and Design. Communications of the ACM 38(9), 36–44 (1995)
13. Wickler, G., Tate, A., Hansberger, J.: Supporting Collaborative Operations within a Coalition Personnel Recovery Center. In: Paper presented at the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, Waltham, MA (2007)
14. Sierhuis, M.: It's not just goals all the way down – It's activities all the way down. In: Engineering Societies in the Agents World, Seventh International Workshop (ESAW 2006). Springer, Dublin (in press, 2006)
15. Minar, M., Burkhart, R., Langton, C.: Swarm Development Group (1996), `http://www.swarm.org`
16. Tatara, E., North, M.J., Howe, T.R., Collier, N.T., Vos, J.R.: An Introduction to Repast Modeling by Using a Simple Predator-Prey Example. In: Agent 2006 Conference on Social Agents: Results and Prospects, Argonne National Laboratory, Argonne, IL (2006)
17. Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.): Multi-Agent Programming: Languages, Platforms and Applications. Springer Science+Business Media, Inc, New York (2005)
18. Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation 2(1), 14–23 (1986)
19. Clancey, W.J., Sierhuis, M., Alena, R., Berrios, D., Dowding, J., Graham, J.S., Tyree, K.S., Hirsh, R.L., Garry, W.B., Semple, A., Buckingham Shum, S.J., Shadbolt, N., Rupert, S.: Automating CapCom Using Mobile Agents and Robotic Assistants. In: American Institute of Aeronautics and Astronautics 1st Space Exploration Conference, Orlando, FL (2005)
20. Sierhuis, M., Diegelman, T.E., Seah, C., Shalin, V., Clancey, W.J., Selvin, A.M.: Agent-based Simulation of Shuttle Mission Operations. In: Agent-Directed Simulation 2007; part of the 2007 Spring Simulation Multiconference, Norfolk, VA, pp. 53–60 (2007)

Progress Appraisal as a Challenging Element of Coordination in Human and Machine Joint Activity

Paul J. Feltovich, Jeffrey M. Bradshaw, William J. Clancey, Matthew Johnson, and Larry Bunch

Florida Institute for Human & Machine Cognition (IHMC)
40 South Alcaniz Street, Pensacola, FL 32502 USA
{pfeltovich,jbradshaw,wclancey,mjohnson,lbunch}@ihmc.us

Abstract. Joint activity, as we define it, is a mutually interdependent social endeavor that requires sufficient predictability among participating parties to enable coordination. Coordination, in turn, sometimes requires the parties to appraise the state of progress of their activities so that, if necessary, they can adjust their actions to meet coordination needs and communicate their status to others as appropriate. A significant impediment as yet precluding the full participation of automation in joint activity with people is its inability to sense and communicate aspects of its state that would allow other participants to meaningfully assess progress toward (or anticipate failure with respect to) mutual objectives. In the current article, we address various issues associated with "progress appraisal" and the challenges it poses for human-machine systems. We point to promising directions for future work.

Keywords: Coordination, culture, human-agent-robotic systems, joint activity, ontology, policy, predictability, regulation, teamwork, progress appraisal, common ground.

1 Introduction

"How are things going?" or "How're you doing?" or even just, "Hello," may seem to be meaningless, perfunctory exchanges among friends and strangers as they pass on the street. But in reality they are more than what they seem. In our ongoing study of interaction among groups of humans and machines, we have come to realize that these deeply-reinforced conventions of decorum serve as simple probes to test the friendliness and predictability of the people in the environment, functioning as rough gauges to the safety of the moment or, on the other hand, to possible cooperative opportunity [23; 43]. With such a perspective in mind, think of what it indicates when, instead of the usual reply of "Fine, thanks," one receives an unaccommodating rebuff, "That's none of your business!"

In situations where people actually *do* join to engage in cooperative work, more complex forms of mutual probes can take on real importance as participants try to assess periodically how the shared work is progressing, especially with regard to interdependent aspects of that work. When machines join people in such endeavors, it

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 124–141, 2008.

is important that they also be able to participate in these ongoing, social processes of probing and appraisal [7, 11, 26, 27].

As automation becomes increasingly pervasive in the ordinary social situations of life, the need for devices capable of such "mixed-initiative" interaction will become ever greater. Although the pursuit of fully autonomous systems is a worthy goal, in many situations coordination with people can improve performance (e.g., [25]). In other situations, agents[1] cannot yet be trusted to perform critical tasks on their own and must be teleoperated by one or more people [12]. Despite these realizations, today's automation is too often implemented as what Sarter, Woods, and Billings call "strong, silent systems" [41] with only two modes: fully automatic and fully manual. In practice this can lead to situations of human "underload," with the humans having very little to do when things are going along as planned, followed by situations of human "overload," when extreme demands may be placed on the human in the case of sudden, unanticipated failure.

This essay explores some of the challenges that *progress appraisal* poses for joint activity involving humans and machines. We first introduce some of the characteristics of social activity and joint activity. Then, in Section 3, we discuss "progress" within human activity. In Section 4, we take up the special role of time, both as a way to specify and to indicate progress. In Section 5, we reach the heart of the matter, as we examine the many factors involved in progress appraisal for humans and machines. These aspirations for machines should be considered *desiderata* rather than current accomplishments. In Section 6, we describe some of our ongoing explorations of these ideas within mixed human-agent-robotic systems, and offer some tentative conclusions.

2 Activity and Joint Activity

Human beings partition the flux of the world into personally meaningful units that researchers have referred to as "activities" [14, 16, 36]. The nature of these activities can usually be revealed by asking a person at any moment, "What are you doing?" The answer will likely be something like "I'm shopping" for a person in a department store, or, from a college professor, "I am giving my lecture in my class," or "I'm preparing my lecture for tomorrow's class." The response will not likely be "I am breathing" or "I am standing on my feet." For observers of conventional scenes involving people doing things, there is usually pretty good agreement about what activity is going on and when one activity has ended and another one has started [37]. Despite such rough consensus in everyday life, what constitutes an activity (as well as its beginning and ending points) is not an objective "thing in the world," but is, rather, a context-sensitive "conceptualization" [14] or "construal" [31, 32] generated by the participants involved. If our professor's attention is absorbed by the painful physical effects he is suffering as a result of his long lecture, he may respond: "I have been standing on my feet too long."

[1] We use the term "agent" in an unqualified way to refer to autonomous or semi-autonomous entities such as software agents, robots, and similar technology.

Previously, we have discussed the nature of *joint activity* as a generalization of that concept proposed by Herbert Clark [20, 34, 35]. Joint activity occurs when people come together to try to accomplish something as a group. We have argued that "The essence of joint activity is interdependence—that what party "A" does depends on what party "B" does, and vice versa (e.g., "One if by land, two if by sea" in Longfellow's account of Paul Revere's famous ride). As soon as there is interdependence, there is often need for coordination in time (e.g., timing a live, multi-party phone call) and/or space (e.g., designating a drop-off point), which in turn requires some amount of predictability and order" [24].

We have also proposed that predictability and order derive from highly diverse regulatory systems that play an essential role in human cultures [23]. These systems can range widely, from devices such as formal law to ethnic folkways to organizational and work norms and practices to informal codes of courtesy and good manners that are meant to govern many aspects of everyday interactions [24]. These can apply to activities (e.g., driving), products (e.g., codes for household electrical wiring), or the social roles (e.g., squad leader) assigned to or adopted by actors.

3 Aspects of Progress

The concept of "progress" itself is, of course, not objective but must be related to the context and aims of individuals. For instance, members of winning and losing baseball teams within a game will have different appraisals of their progress toward victory. On the other hand, they will be more likely to share similar views with respect to clocks and landmarks (e.g., innings) that serve as indicators of the remaining time left to play. Two members of the same team may, in turn, appraise progress differently because they are concerned with their own piece of the overall joint activity in addition to the more global perspective. A good example of discrepancy in appraisal comes from a commentary on the ethnic turmoil in the Balkans, showing how different stakeholders can judge progress differently—in this case by attending to different components:

> The omens before the March pogrom did not all auger ill. Political officers in the UN mission who had been monitoring inter-ethnic tolerance were seeing progress. Returns of displaced Serbs had increased… Talks between Belgrade and Pristina about a variety of practical issues of mutual interest had recently begun… Minority representation in the Kosovo Police Service had improved… But there were also signs that tensions were reaching breaking point… The Kosovo Assembly, the territory's elected parliament, had marginalized the significant number of minority members… The official opening of the Assembly's refurbished hall was marred by Kosovo Serb's understandable complaints about murals depicting scenes that reflected only the Albanian's view of history… [33, p. 7]

If the world could be completely objective and predictable, there might be little need to monitor the progress of the multiple efforts involved in a joint activity. Consider, for instance, two individuals who are involved in independent activities for the

afternoon (estimating the time needed for all of them) and then plan to meet at a certain place and time that evening. If everything goes as expected, they will simply meet at the appointed place and time, without having to consider their intermediate progress. But disruptions of various kinds often impinge in such situations (e.g., one of the parties is delayed by traffic or an emergency at home), and so, to successfully accomplish their coordination aim, they will need to communicate their status at critical junctures and adjust accordingly.

We note that even assessing one's own status regarding progress can be challenging. In the case of machines, though they may have excruciatingly detailed access to aspects of their internal functions, they are typically oblivious to the future implications of their current state or to important aspects of the world around them that may affect their ability to perform. It has long been a complaint about automation that it cannot make or broadcast these kinds of high level, reflective judgments and that this limitation often causes situations to unravel helplessly, as machines fail without warning or report trouble too late to enable corrective measures [5, 13, 39, 41]. This article addresses the kinds of requirements that must be met if machines were to have significant capability for *judging their own progress and, perhaps, being able to offer reliable warnings regarding deviations from expected progress or, in the worst case, an advance notice of impending failure.* Time is an important dimension for these kinds of assessments—a topic we take up next.

4 Time as a Special Dimension

Time serves many roles as part of progress appraisal. For instance, it can function as constraint ("I have—or estimate that I have—thirty minutes to complete the task"), a planning factor ("Given the time I have, I will try doing the task this particular way"), or more purely as an indicator of progress ("This is taking way too long, I'm not going to make it in time").

Time, along with the degree of tightness in the coupling of the interdependent activities, is often a central factor in judging progress. In general, shorter time and tighter coupling reduce the margins of error in joint activities and allow less time for appraisal and the making of any necessary adjustments. In any form, progress appraisals will need to be made in different kinds of contexts that involve timing:

- **Deterministic: Designed, Fixed Timing.** These situations involve fixed deadlines that a process must meet, such as a final date for submitting a research proposal or making a reservation. These deadlines are set in order to coordinate with the processes of others (e.g., the review and handling procedures by the respective organizations involved) in a fashion that has been specifically designed to minimize surprises and to allow the deadlines (coordination points) to be planned, stable, and public. In some instances adjustments are possible, as in extensions, but there is often fixed procedure (and set times) for making these, too. A key characteristic of such situations is the desire for predictability (and relative stability) of both the processes involved in the coordination and the timing of the interdependencies among the parties. For example, sub-process relations, causal influences, and causal effects are deterministic.

- **Emergent Internal Relations: Contingent Inter-Process, Fluid Timing.** In this kind of coordination, a deadline is not fixed as a time, but, rather, is dependent on the progress of other interdependent processes (see section on "Other activities" below). For example, a coauthor on a scientific article cannot make progress on her part of the writing until some data are analyzed by another colleague (with both authors, perhaps, subject to a fixed deadline for completing their overall project). In such cases, both interdependent processes must somehow be made aware of (or estimate) progress within the other. When all goes "as planned," they may coordinate through a pre-set schedule or plan. But perhaps more often, they will need to communicate their progress to one another as they move forward.
- **Emergent External Influences: Timing Imposed from Indirect Influences.** Many influences external to the main joint activity affect the efficiency of its completion. For example, available resources may be germane, as when a pilot is waved off from a scheduled landing site and needs to find an emergency site that is reachable with his available fuel. Loss of primary communication modes slows (or prevents) interdependent progress. Local regulations may impinge, unanticipated obstacles (or affordances) pop up, the weather change, and so forth. *Surprises*, in general, seem particularly relevant in this category.
- **Emergent External Effects: Timing Imposed by the Half-life (Perishability) of the Usefulness of Components.** Elements may only "work" for some purpose for a limited window of time. For example, a photograph of an active battlefield, provided by a higher command to an officer on the ground, will likely have a shorter span of usefulness than a weather report pertaining to the same locale, or an even more stable geological survey. The value of work performed, or information provided, may plummet to zero if it does not appear until after the "usefulness deadline" has passed. More positively, partial products will sometimes suffice for the coordinating party to be able to make *some* progress, even if less than the full product would allow. The parties involved have a number of options if they determine that a usefulness deadline may be missed. For example, they can intervene by providing more resources to the activity, or they can seek the product from an alternative source.

To summarize, the necessary timing of a process (always subject to need for adjustment) is determined in advance (a schedule) or emergent. Emergent timing is contingent on sub-process relations of variable duration and on external factors. External factors include 1) *causal influences on the process* that may change before the process is complete and 2) *causal effects of the process* (e.g., functional value of its products) whose timing affects the external factors. Because timing is inherent in causal processes—whether parallel or sequential—the temporal nature of the process provides information for specifying or indicating how the process is progressing.

5 Factors Affecting Progress and Progress Appraisal

Many other factors affect progress and its appraisal in interdependent human-machine systems. We will now discuss some of these and give examples to clarify the nature of the factors and to provide ideas about how they might be addressed. While the particular factors presented have been stimulated by some of our own experiences in

developing human-agent-robotic systems over many years, as well as those reported by others (e.g., the criticality of communications), we make no claim to exhaustiveness. In this sense, the factors discussed can function as a stimulus to further investigation and reporting.

Communication. Coordination without some form of communication is challenging for humans and agents alike. In critical situations with a high degree of mutual interdependence, a loss of communication can spell danger. This has been a major lesson from almost all recent international responses to disaster, as reflected in pleas, contained in many after-action reports, for *"bullet-proof" and interoperable* communication devices and networks (e.g. [22]).

Beyond the mere availability of communication, the quality and timeliness of progress reports play a key role. Reports that arrive later than planned because of a downed system can handicap the activities of others who are dependent on them. The negative effect often ripples outward in a variety of ways, impacting available time to adjust or to consult with others to get assistance. Messages that deceptively or inaccurately report progress have similar effects. Over-messaging can also have deleterious effects on progress and its appraisal [40].

***Human Example*:** Joe and a colleague, who will arrive first at the airport, agree to meet at the AJAX car rental desk where they will share a ride. Upon arriving at "the" desk, Joe discovers that this particular airport has four AJAX desks, one in each of three terminals and one outside the terminal. His partner is not there. He tries to call his partner from his cell phone but finds that he has no signal at this spot. After some time, he decides to ask the agent at the nearest AJAX desk to call the other desks to look for his partner. They finally connect and successfully co-locate.

***Agent Teamwork Example*:** A human-agent team could incorporate a policy that causes regular checks on all agents. If they do not respond, one knows that either the communication has failed—or worse. At a specified heartbeat rate, a probe assigned to each agent could automatically test capability without interfering with the ongoing work of the agents (cf. [2]). Also desirable would be a system of back-up communication that could be engaged when first-line devices fail or are suspect, and an agreed upon "safe" mode or default behavior to which each agent would revert if communication becomes impossible.

Landmarks. These are recognizable entities, including partial products, that should be seen or produced if acceptable progress is being made.

***Human Example*:** Knowing the general distance from the airport to his hotel and the current flow rate of traffic, Joe believes that he should have seen the bridge on his map by now. He becomes concerned that he may have made a wrong turn somewhere and starts comparing the street signs he can see around him with street names on the map. Having determined his current location, he adjusts his route accordingly.

***Agent Teamwork Example*:** Landmarks encountered, goals achieved, or results produced during a process can be indicators of current state in comparison to observables, that is, to detectables known to reflect degree of progress toward goal states. Physical metrics, such as distance from a physical destination, are one kind of indicator. But it is also important to be able to track more abstract indicators. For

example, if there is a simple plan that contains two steps, it would count as progress when the first step was completed.

Successful use of landmarks clearly depends on familiarity with an activity. One needs to have expectations of what kinds of things appear or are revealed, get produced, or are consumed, as a process develops to completion. This can be supported by actual experience and learning, or by various sorts of props that represent vicarious experience (e.g., maps, plans, guides, checklists). Examples of landmarks include goals achieved or steps completed in a plan, entities encountered or revealed in relation to those expected, computational results produced in comparison to those required, resource use in relation to average consumption, and actual time intervals between two events in comparison with "normal" intervals.

Helps, Obstacles, and Affordances. By these we mean all the diverse elements that aid, hinder, or merely allow progress. Some of these may be of long standing, while others pop-up unexpectedly (e.g., summer-long bridge construction vs. a traffic accident).

***Human Example*:** On his drive from the hotel to the airport, Joe discovers that his intended exit from the highway has been closed that day for repairs. He checks his map to see how far he is from the next exit.

***Agent Teamwork Example*:** Policy can require agents to notify other team members of helps, obstacles, and potential affordances that will affect their individual performance—or that of the team. Such an obligation recalls the teamwork heuristic, developed by Cohen and Levesque, that required agents to tell team members when a team goal was achieved or became impossible or irrelevant [21]. This generic approach saved developers from having to write numerous special purpose exception handling procedures for specific situations [44]. Degree of anticipation is an important factor in dealing with helps, obstacles, and affordances. Anticipated obstacles, e.g., increased crowds at restaurants around lunch hour, can be planned for (block out more time) or worked around (go earlier of later). Accuracy in anticipation, again, depends on familiarity and learning.

Resources. Resources range from such things as *energy and bandwidth* to necessary *artifacts and tools* (e.g., a hammer, a car [and energy for the car], a map). What stands as a resource is relational, that is, "This A serves as a resource for this B in context C." Thus, just about anything can serve as a resource for something else in the right context.

***Human Example*:** Joe runs out of gasoline in his car and stalls. He tries to use his cell phone to call for help, but he discovers his phone battery is dead.

***Agent Teamwork Example*:** Levels of necessary consumables and achievability of enabling conditions for actions can be monitored. For example, batteries, communications, and other resources can often be monitored—and perhaps reasoned about (e.g., "Do I have enough gasoline to make it to my destination?"). Enabling conditions for the execution of actions can also be investigated (e.g., "Do I have a hammer if I need to pound a nail, a vehicle if I need to make a trip?").

Knowledge. Included here are basic knowledge of requirements for an activity and also the means for addressing these, including alternative means, routes, geography, places to acquire consumable resources (e.g., gas and food), people who can help, the

roles of team members, knowledge of pertinent regulations, access routes, and so forth. Experience with a joint activity aids progress appraisal in at least two ways. First, with time the parties involved build up norms for how much time component activities usually take and the likely impediments that may arise. This helps detection of unusual time delays and allows preplanning and workarounds for many known potential impediments. Second, with experience, team members get to know each other's roles, habits and manners, for example, leadership, timeliness, trustworthiness, work-habits, and degree of communication availability. If a team facing a hard deadline needs a deliverable from a team member by a certain time, they will appraise the situation differently if the partner is reliable and always comes through, versus another colleague whose delivery patterns are spotty.

***Human Example*:** Joe, a visitor, runs out of cash in Japan and, to no avail, tries to find ATM machines or banks to help him. He does not know that in Japan many of the functions carried out by banks in other countries are, instead, handled by post offices. A local resident tries to explain this situation to Joe, but Joe cannot understand Japanese.

***Agent Teamwork Example*:** Agent planning capabilities generally address problems concerning what knowledge, actions, and resources are necessary to complete a given task. Typically, however, such planners are limited in their ability to be self reflective, let alone being able to reason effectively about coordination issues involving other human and agent team members. Our research group is working on collaborative planning approaches that take as a premise that people are working in parallel alongside (more or less) autonomous systems, and, hence, adopt the stance that the processes of understanding, problem solving, and task execution are necessarily incremental, subject to negotiation, and always tentative [1, 6]. In this way, agents and robots can work with people in a mixed-initiative mode—doing as much as they can autonomously based on their own knowledge, but also being aware of when they need to take direction or ask for help from others.

Mistakes. These are actions other than those intended, including "slips" [15, 38]. When recognized, mistakes should trigger attempts to recoup and re-estimate progress (e.g., how much time, relative to the previously expected time, the process may now take for completion, given the mistake).

***Human Example*:** Joe arrives at his departure gate for a connecting flight. Just as the final call is being made, he considers whether he has time to visit the restroom. He decides he has just enough time. Upon exiting the restroom, he turns down the hall and starts walking. Suddenly, he realizes he went the wrong direction. Since he caught the mistake quickly, he turns around and walks at a regular pace. Had he walked farther before noticing the error, he may have had to quicken his step.

***Agent Teamwork Example*:** Unlike obstacles, mistakes are the result of the agent's own choices. Effects of these choices can be monitored. When a mistake is detected, the agent can abandon the plan, retrace back to the intended path and then continue, ask for help, or construct a new path. Teng (IHMC) has developed an initial version of *Kab* (*KAoS abstract backup*), a new special-purpose planner that works in conjunction with the KAoS *Kaa* component (see below) to help agents and agent teams formulate

and select appropriate generic backup plans for such situations. Unlike typical approaches for this problem, *Kab* relies on a small number of human-compatible strategies for plan repair based on our observations of teamwork-in-practice, rather than a collection of general-purpose formalisms grounded in logic alone.

Regulatory Devices. These include any rules, regulations, customs, or other constraints (and affordances) that apply to the activity at hand (e.g., speed limits, rights of way, policies) or to the roles of team members (e.g., associated restrictions, rights, obligations).

***Human Example*:** Not having planned well, Joe finds himself with less than ample time to get to the airport. On the map, he sees an alternative route that looks shorter and more direct. He takes this route and soon discovers that the speed limit on this road is low. He also reads on a sign that the fines for speeding at this particular time are doubled because there are workers doing maintenance on the road. He considers tracing back to his original route or trying to find yet another one.

***Agent Teamwork Example*:** This involves the detection of impedance to progress due to the enforcement of a regulatory mechanism (e.g., a policy, rule, or role responsibility/obligation). To the extent these situations can be represented by (computational) policy, this is straightforward (e.g., having a policy to prevent entry to a restricted area). *Kaa* is an adjustable autonomy capability that uses decision-theoretic reasoning to determine how to adjust policies, resources, or circumstances to assist in the achievement of team objectives [9].

Conflicts. Because resources such as time and attention are finite, conflicts regarding their allocation sometimes occur. These include conflicts among alternative activities, goals, obligations, and allegiances.

***Human Example*:** Not only must Joe board his flight in time, he also needs to finish off a piece of writing before boarding (finishing it after the flight would be too late) to send to a colleague back home who is completing submission of a grant proposal under deadline. He decides to move close to the airplane entrance door, not board when first called, and work on the writing up until the last moment before the airplane door is closed.

***Agent Teamwork Example*:** Conflicts can be handled by agents in several ways. One is to split the team, if possible, to cover the different duties (differential reassignment). For example one agent covers one task, and another covers the other. Their work can also be prioritized—a kind of triaging. Alternatively, they might do a merely adequate, rather than a superb, job on each task so they create time for both. They might just speed up. Policies can specify priorities for competing demands and can also constrain what alternatives are available for re-tasking and delegation.

Changes of Plans. Sometimes in the midst of the conduct of a planned joint activity, events evolve that cause overall aims to change. This may happen for many reasons, for example, a more pressing need/objective has emerged, elements of the current operation have failed so badly that the original aim must simply be aborted, or an opportunity has arisen that enables achievement of a higher priority goal. The effect on progress will depend on how discrepant the new objective and plan are from the original one.

***Human Example*:** Joe is in a distant town for a three-day academic workshop. At the end of the first day, he receives a call from his wife back home reporting that there has been a family emergency. He immediately starts making arrangements to return home, recruiting a colleague to deliver his paper at the workshop.

***Agent Teamwork Example*:** The topic of this essay, progress appraisal, is pertinent to judging whether an activity is progressing acceptably, with change of plan being a recurrent theme (e.g., as the result of a surprise). In that sense, the whole article involves suggestions for building this capability in agents. One approach utilizes "back-up" plans, both contingency (worked out in advance) plans and dynamic re-planning (see brief discussion of the development of *Kab*, above).

Other Activities. This category is particularly pertinent to interdependent activities. We are interested in *joint* activity, so how well one activity "is going" is not independent to that activity, but is contextually related to how other processes are progressing, especially regarding their points of interdependency (e.g. how fast one participant must complete its activity depends on how fast some other process needs the output to accomplish *its* part of the joint activity).

***Human Example*:** At their destination airport, Joe and a colleague, who is coming from a different city but is scheduled to arrive about the same time as Joe, are to join each other and to share a rental car to their hotel. Joe's flight experiences a moderate delay in departure, and he calls his friend to tell him his flight will be late. Because the delay will be relatively short, the friend says he will get something to eat at the airport and wait for Joe. Had the estimated delay been long, the friends might have decided just to make their own ways to the hotel.

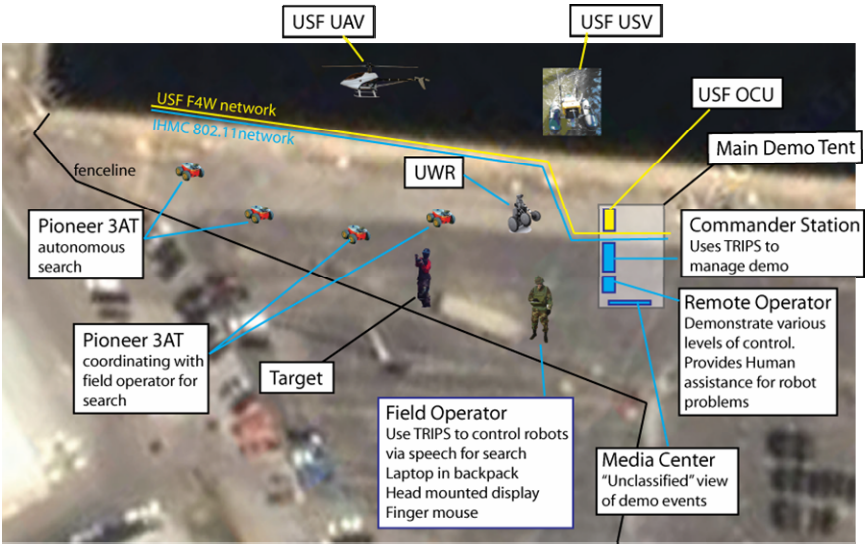


Fig. 1. Overview of Coordinated Operations Exercise Components

***Agent Teamwork Example*:** Any evidence, reported by an agent or solicited from an agent, indicating that some process in which it is involved is progressing differently from expected, especially with reduced progress, is important. One approach is for agents involved in joint activity to monitor each other's progress and to make adjustments as needed along the way.

6 Applications to Human-Agent-Robotic Teamwork

Our group is applying the ideas presented in this article to facilitate joint activity in mixed human-agent-robot teams.[2] We have developed the KAoS HART (Human-Agent-Robot Teamwork) Services framework as a means to flexibly specify, analyze, enforce, and adapt policy constraints to facilitate team coordination [8, 45]. KAoS policies, represented in OWL (W3C's Web Ontology Language, http://www.w3.org/2004/OWL), are of two types: *authorizations* (constraints that permit or forbid some action in a given context) and *obligations* (constraints that require some action or waive a pre-existing requirement). More complex teamwork policies are built out of these primitive types. To more adequately represent some of the subtleties of joint action, we are augmenting the policy ontology with a broader Ontology of Regulation [24].

In the context of human-agent-robot teams, we have been involved in domains such as space exploration, disaster response, and military operations [9, 17, 18, 28, 42]. A recent field exercise, particularly germane to the present essay, involved complex coordinated operations of a team consisting of two humans and seven robots (Fig. 1) [30]. In this demonstration, the team had to perform reconnaissance of a port facility to determine the presence of underwater obstacles, explosives, structural soundness of pier facilities, and the nature and extent of any armed resistance. In the course of this surveying, an intruder was detected, and the team then needed to secure the boundaries to prevent escape, and to search the area to find and apprehend the intruder. The team consisted of two humans: a remote commander and a lieutenant in the field interacting locally with the robots. The robots included an unmanned air vehicle, an unmanned water surface vehicle, a highly mobile IHMC "tBot" robot, and four Pioneer 3AT ground vehicles, variously equipped with sonar, GPS, cameras, and SICK lasers.

6.1 Progress Appraisal in the Coordinated Operations Exercise

Members of the human-robot teams were strongly associated with roles. Roles can be thought of as ways of packaging rights and obligations that go along with the necessary parts that people play in joint activities [10, 29]. Knowing one's own role

[2] We realize that there are important differences between human teams and the mixed teams of which we write. Even the authors of this paper have had lively debate as to whether the use of the term "team" is appropriate in this context and whether machines and software can appropriately be classed as "team members." While recognizing the significant—and perhaps insurmountable—differences between the contributions that technology and people can make to joint activity, a large segment of the research community uses "team" as a rough way of characterizing the ideal forms of interaction to which we aspire. For snapshots of this ongoing debate, see [3, 4, 18].

and the roles of others in a joint activity establishes expectations about how others are likely to interact with us, and how we think we should interact with them. In addition, progress is often associated with the duties of a role. When roles are well understood and regulatory devices are performing their proper function, observers are likely to describe the activity as highly-coordinated. On the other hand, violations of the expectations associated with roles and regulatory structures can result in confusion, frustration, anger, and a breakdown in coordination.

Collections of roles are often grouped to form organizations such as teams. In addition to regulatory considerations at the level of individual roles, organizations themselves may also add their own rules, standards, traditions, and so forth, in order to establish a common culture that will smooth interaction among parties.

The lesson here for mixed human-agent-robot teams is that the various roles that team members assume in their work must include more than simple names for the role and algorithmic behavior to perform their individual tasks. They must also, to be successful, include regulatory structures that define the additional work of coordination associated with that role.

Consistent with this role-based orientation, coordination of search and apprehension activities was facilitated by five sets of KAoS policies addressing chain of command, acknowledgement, and progress appraisal issues as they relate to the requestor, the team leader, and other team members. The particular policy sets chosen for initial implementation are related to core components of our developing Progress Appraisal investigations. Some are directly related to progress appraisal as indicated by their titles. Two others, Chain of Command and Acknowledgement, are important auxiliary components. The first puts into play a central set of regulatory constraints that can aid or impede progress in our particular domain of application, a military operation (see section on "Regulatory Devices" above). The latter is a basic set of communication policies for supporting the basic progress appraisal process (see section on "Communications" above). We note the similarities of some aspects of our work to Winograd's Coordinator [46].

6.1.1 Chain of Command

This policy set enforces a hierarchical, military-style chain of command and consists of four policies:

- A Robot is not authorized to perform Action requests from just any Requestor EXCEPT
- A Robot is authorized to perform Actions requested by its Team Leader
- A Robot is authorized to Accept Actions requested by a higher authority
- A Robot is authorized to Accept Actions that are self-initiated

These policies support the norms of authority in a military operation. Here, we focus on the role of a leader in relationship to subordinates (e.g., the robots, above).

6.1.2 Acknowledgment

This policy set enforces acknowledgment of all commands, except those that are directly observable:

- A Robot is obligated to Acknowledge when the Robot Accepts an Action EXCEPT
- A Robot is not obligated to Acknowledge Teleoperation requests
- A Robot is not obligated to Acknowledge Query requests

"Acknowledgement" means, "I (e.g., a robot) got your message," *prior to acting on it*. Simply acting on it, when the act's execution is visible, usually obviates the need for pre-acknowledgement, as Clark has written about regarding his "Joint Action Ladder" [20].

6.1.3 Requested Action Progress Appraisal

This policy set enforces communication norms between a requestor and requestee, based on progress:

- For *Continuous Actions*, A Robot is obligated to notify the requestor when the Status of the requested Action *changes*
- A Robot is obligated to notify the Requestor when requested Action is Finished (includes statuses of Completed, Aborted, and Failure)

EXCEPT *Certain types of commands are directly observable and do not require feedback unless something goes wrong.*

- A Robot is not obligated to notify the Requestor when a requested Teleoperation Action is Completed successfully
- A Robot is not obligated to notify the Requestor when a requested Query Action is Completed successfully

These help provide the requestor with progress appraisal information *for execution of the specific action requested,* except when progress can be assessed more directly, e.g. can be seen.

6.1.4 Leader Progress Appraisal

This policy set enforces communication norms with a team leader based on progress:

- A Robot is obligated to notify its Team Leader when an Action is requested by a higher authority (higher than the team leader)
- A Robot is obligated to notify Its Team Leader when starting a self-initiated Action
- A Robot is obligated to notify its Team Leader when a self-initiated Action is Finished (includes statuses of Completed, Aborted, and Failure)

These policies serve much like those of 6.1.3, except that the role of "leader" demands some special kinds of notices, compared to a general Requester (role).

6.1.5 Group Peer Progress Appraisal

This policy set enforces communication norms between members on the same team based on progress:

- A Robot is obligated to notify other participants in a Joint Task when the Joint Task is Finished (includes statuses of Completed, Aborted, and Failure)

- A Robot is obligated to notify its Team Members if the Team Goal is Aborted and no longer applicable

Again, these policies support appraisal of progress among members of the working group, but in some ways differently from the like support provided to a "Leader (role)."

6.2 Further Policy Considerations

Each of the operative policies helps to maintain common ground (mutual understanding) by enabling the robotic agents to coordinate with their human counterparts in a manner consistent with human norms and expectations [19, 34, 35]. As a simple example, when asked to perform a task, the robots will acknowledge the request (simply saying that it has *received* the message; acknowledgement in this sense does not refer to its possible subsequent actions taken). However, for teleoperation, the requests are numerous and the effect, the resulting action, is directly observable, so acknowledgement would become annoying and detrimental. Therefore we waive this requirement. Similarly, if the lieutenant is in charge of a robot, and his commander overrides his authority and tasks the same robot, we ensure the lieutenant is informed, rather than leave the lieutenant in a state of confusion about the unexpected actions of the robot.[3] Another example of progress appraisal is that when a robot is tasked to search for something and it finds (or loses) it, the robot tells the requestor of this status change—something obvious to humans but typically not considered explicitly in robot operations.

We have previously noted the need to design-in collaborative capabilities in robots at a more basic level than is typically done [28]. One area we are focusing on for the future is determining how to code robots in a manner that allows for a finer grained progress appraisal. It would be useful, not only to know if an action is completed or aborted, but also if the robot is struggling or delayed. We have been working on several examples of robotic behavior where we can provide this type of information.

A final challenge is dealing with the more subjective aspects of progress appraisal (see Section 3). We note, for example, the difference in difficulty between assessing progress on a more objective task that depends on closing a known distance between a robot and its target and a more subjective task that depends on aggregating a number of imperfectly known estimates, perhaps even reflecting different points-of view or conceptualizations of the meaning of the activity. This will be a daunting problem [24].

7 Conclusions

Complex operations involving mixed teams of humans, software agents, and robots, require strong tools to support coordination of the interdependencies among the activities of the components. Since, in the real world, activities usually do not play out exactly as expected, successful coordination often requires that adjustments in planned activities be made to accommodate disturbances in progress within the

[3] The Commander has higher authority in this case; extant policy automatically detects the conflict in commands and de-conflicts according to chain of command.

coordinating activities. This, in turn, requires that agents have an understanding of their progress so that they can convey this to other participants. Historically, this kind of assessment of progress (with associated warnings of impending trouble) has been a critical limitation of automation, and this lack has contributed to some automation disasters. Our research group is confronting this specific kind of limitation. The work is just beginning, but we are optimistic that important progress can be made.

Acknowledgements

Our thanks to the following individuals for their substantial contributions: James Allen, Maggie Breedy, Marco Carvalho, Tom Eskridge, Lucian Galescu, Hyuckchul Jung, James Lott, Robin Murphy, Jerry Pratt, Anil Raj, Signe Redfield, Richard Smith, Niranjan Suri, Choh Man Teng, and Andrzej Uszok. The research was supported in part by grants and contracts from the U.S. Air Force Office of Scientific Research to Dartmouth College through subcontract (5-36195.5710), the Office of Naval Research (N00014-06-1-0775), and the U.S. Army Research Laboratory through the University of Central Florida under Cooperative Agreement Number W911NF-06-2-0041. William J. Clancey has been supported in part by funding from NASA's Constellation Program. The article does not necessarily reflect the views of any of these agencies. We also thank the fine reviewers and the editors of this volume.

References

1. Allen, J.F., Ferguson, G.: Human-machine collaborative planning. In: Proceedings of the NASA Planning and Scheduling Workshop, Houston, TX (2002)
2. Arkin, R.C.: Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments. Journal of Robotic Systems 9(2), 197–214 (1992)
3. Biever, C.: Bots as peers: It's all a matter of teamwork. San Francisco Chronicle, San Francisco, CA, May 6 (2007) (accessed September 15, 2007), http://sfgate.com/cgi-bin/article.cgi?f=/c/a/2007/05/06/ING9GPK9U71.DTL
4. Biever, C.: If you're happy, the robot knows it. The New Scientist 193(2596), 30–31 (March 24, 2007) (accessed September 15 (2007), http://technology.newscientist.com/article/mg19325966.500-if-youre-happy-the-robot-knows-it.html
5. Bradshaw, J.M., Sierhuis, M., Acquisti, A., Feltovich, P., Hoffman, R., Jeffers, R., Prescott, D., Suri, N., Uszok, A., Van Hoof, R.: Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications. In: Hexmoor, H., Falcone, R., Castelfranchi, C. (eds.) Agent Autonomy, pp. 243–280. Kluwer Academic Publishers, Dordrecht (2003)
6. Bradshaw, J.M., Acquisti, A., Allen, J., Breedy, M.R., Bunch, L., Chambers, N., Feltovich, P., Galescu, L., Goodrich, M.A., Jeffers, R., Johnson, M., Jung, H., Lott, J., Olsen Jr., D.R., Sierhuis, M., Suri, N., Taysom, W., Tonti, G., Uszok, A.: Teamwork-centered autonomy for extended human-agent interaction in space applications. In: AAAI 2004 Spring Symposium, 22-24 March, 2004. Stanford University, CA (2004)

7. Bradshaw, J.M., Feltovich, P., Jung, H., Kulkarni, S., Taysom, W., Uszok, A.: Dimensions of adjustable autonomy and mixed-initiative interaction. In: Nickles, M., Rovatsos, M., Weiss, G. (eds.) AUTONOMY 2003. LNCS (LNAI), vol. 2969. Springer, Heidelberg (2004)
8. Bradshaw, J.M., Feltovich, P.J., Jung, H., Kulkarni, S., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Johnson, M., Sierhuis, M., Taysom, W., Uszok, A., Van Hoof, R.: Policy-based coordination in joint human-agent activity. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, The Hague, Netherlands, October 10-13 (2004)
9. Bradshaw, J.M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Suri, N., Taysom, W., Uszok, A.: Toward trustworthy adjustable autonomy in KAoS. In: Falcone, R. (ed.) Trusting Agents for Trustworthy Electronic Societies. Springer, Berlin (2005)
10. Bradshaw, J.M., Feltovich, P., Johnson, M., Bunch, L., Breedy, M.R., Jung, H., Lott, J., Uszok, A.: Coordination in Human-Agent Teamwork. Invited Paper and Presentation. In: AAAI Fall Symposium, November 8-10 (2007)
11. Bruemmer, D.J., Marble, J.L., Dudenhoeffer, D.D.: Mutual initiative in human-machine teams. In: Proceedings of the 2002 IEEE 7th Conference on Human Factors and Power Plants, vol. 7, pp. 22–30 (2002)
12. Burke, J.J., Murphy, R.R., Coovert, M., Riddle, D.L.: Moonlight in Miami: A field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. Human-Computer Interaction 19(1-2), 85–116 (2004)
13. Christofferson, K., Woods, D.D.: How to make automated systems team players. In: Salas, E. (ed.) Advances in Human Performance and Cognitive Engineering Research, vol. 2. JAI Press, Elsevier (2002)
14. Clancey, W.J.: Situated Cognition: On Human Knowledge and Computer Representations. Cambridge University Press, Cambridge (1997)
15. Clancey, W.J.: Conceptual Coordination: How the Mind Orders Experience in Time. Lawrence Erlbaum, Hillsdale (1999)
16. Clancey, W.J.: Simulating activities: Relating motives, deliberation, and attentive coordination. Cognitive Systems Review (2002);Special issue on Situated and Embodied Cognition
17. Clancey, W.J.: Automating Capcom: Pragmatic operations and technology research for human exploration of Mars. In: Cockell, C. (ed.) Martian Expedition Planning. AAS Science and Technology Series, vol. 107, pp. 411–430 (2004)
18. Clancey, W.J.: Roles for agent assistants in field science: Understanding personal projects and collaboration. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews 32(2) (2004)
19. Clancey, W.J., Sierhuis, M., Damer, B., Brodsky, B.: Cognitive modeling of social behavior. In: Sun, R. (ed.) Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge University Press, New York City (2005)
20. Clark, H.H.: Using Language. Cambridge University Press, Cambridge (1996)
21. Cohen, P.R., Levesque, H.J.: Teamwork. SRI International, Menlo Park (1991)
22. Davis, L.E., Rough, J., Ceccine, G., Gareban-Schaefer, A., Zeman, L.L.: Hurricane Katrina: Lessons for Army Planning and Operations. Rand Corporation, Santa Monica (2007)
23. Feltovich, P., Bradshaw, J.M., Jeffers, R., Suri, N., Uszok, A.: Social order and adaptability in animal and human cultures as an analogue for agent communities: Toward a policy-based approach. In: Omacini, A., Petta, P., Pitt, J. (eds.) Engineering Societies for the Agents World IV. LNCS, vol. 3071, pp. 21–48. Springer, Berlin (2004)

24. Feltovich, P., Bradshaw, J.M., Clancey, W.J., Johnson, M.: Toward and Ontology of Regulation: Socially-based Support for Coordination in Human and Machine Joint Activity. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2007. LNCS (LNAI), vol. 4457, pp. 175–192. Springer, Heidelberg (2007)
25. Fong, T.W., Thorpe, C., Baur, C.: Robot as partner: Vehicle teleoperation with collaborative control. In: Workshop on Multi-Robot Systems, Naval Research Laboratory, Washington, DC (March 2002)
26. Fong, T.W., Nourbaksh, I., Ambrose, R., Simmon, R., Scholtz, J.: The peer-to-peer human-robot interaction project. In: AIAA Space 2005 (September 2005)
27. Goodrich, M.A., Olsen Jr., D.R., Crandall, J.W., Palmer, T.J.: Experiments in adjustable autonomy. In: Proceedings of the IJCAI_01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents, Seattle, WA (August 2001)
28. Johnson, M., Bradshaw, J.M., Feltovich, P., Jeffers, R., Uszok, A.: A semantically-rich policy-based approach to robot control. In: Proceedings of the International Conference on Informatics in Control, Automation, and Robotics, Lisbon, Portugal (2006)
29. Johnson, M., Feltovich, P.J., Bradshaw, J.M., Bunch, L.: Human-robot coordination through dynamic regulation. In: Proceedings of the International Conference on Robotics and Automation (ICRA), Pasadena, CA (in press, 2008)
30. Johnson, M., Intlekofer Jr., K., Jung, H., Bradshaw, J.M., Allen, J., Suri, N., Carvalho, M.: Coordinated operations in mixed teams of humans and robots. In: Marik, V., Bradshaw, J.M., Meyer, J. (eds.) Proceedings of the First IEEE Conference on Distributed Human-Machine Systems (DHMS 2008), Athens, Greece (in press, 2008)
31. Kelly, G.A.: The Psychology of Personal Constructs. Two vols. Norton, New York (1955)
32. Kelly, G.A.: A Theory of Personality. W. W. Norton & Co, New York City (1963)
33. King, I., Mason, W.: Peace at Any Price: How the World Failed Kosovo. Cornell University Press, Ithaca (2006)
34. Klein, G., Feltovich, P.J., Bradshaw, J.M., Woods, D.D.: Common ground and coordination in joint activity. In: Rouse, W.B., Boff, K.R. (eds.) Organizational Simulation, pp. 139–184. John Wiley, New York City (2004)
35. Klein, G., Woods, D.D., Bradshaw, J.M., Hoffman, R., Feltovich, P.: Ten challenges for making automation a team player in joint human-agent activity. IEEE Intelligent Systems 19(6), 91–95 (2004)
36. Leont'ev, A.N.: The problem of activity in psychology. In: Wertsch, J.V. (ed.) The Concept of Activity in Soviet Psychology. M. E. Sharpe, Armonk (1979)
37. Newtson, D.: Attribution and the unit of perception of ongoing behavior. Journal of Personality and Social Psychology 28, 28–38 (1973)
38. Norman, D.A.: Categorization of action slips. Psychological Review 88, 1–15 (1981)
39. Norman, D.A.: The 'problem' with automation: Inappropriate feedback and interaction, not 'over-automation'. In: Broadbend, D.E., Reason, J., Baddeley, A. (eds.) Human Factors in Hazardous Situations, pp. 137–145. Clarendon Press, Oxford (1990)
40. Patterson, E.S., Watts-Perotti, J., Woods, D.D.: Voice loops as coordination aids in Space Shuttle Mission Control. Computer Supported Cooperative Work 8, 353–371 (1999)
41. Sarter, N., Woods, D.D., Billings, C.E.: Automation surprises. In: Salvendy, G. (ed.) Handbook of Human factors/Ergonomics, 2nd edn. John Wiley, New York (1997)
42. Sierhuis, M., Bradshaw, J.M., Acquisti, A., Van Hoof, R., Jeffers, R., Uszok, A.: Human-agent teamwork and adjustable autonomy in practice. In: Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Nara, Japan, May 19-23 (2003)

43. Smith, W.J.: The biological bases of social attunement. Journal of Contemporary Legal Issues 6, 361–371 (1995)
44. Tambe, M., Shen, W., Mataric, M., Pynadath, D.V., Goldberg, D., Modi, P.J., Qiu, Z., Salemi, B.: Teamwork in cyberspace: Using TEAMCORE to make agents team-ready. In: Proceedings of the AAAI Spring Symposium on Agents in Cyberspace, Menlo Park, CA (1999)
45. Uszok, A., Bradshaw, J.M., Johnson, M., Jeffers, R., Tate, A., Dalton, J., Aitken, S.: KAoS policy management for semantic web services. IEEE Intelligent Systems 19(4), 32–41 (2004)
46. Winograd, T.: A language/action perspective on the design of cooperative work. Human-Computer Interaction 3(1), 3–30 (1987-1988)

Automated Web Services Composition with the Event Calculus*

Onur Aydın[1], Nihan Kesim Cicekli[2], and Ilyas Cicekli[3]

[1] Microsoft Corporation, Seattle, USA
[2] Department of Computer Engineering, METU, Ankara, Turkey
[3] Department of Computer Engineering, Bilkent University, Ankara, Turkey
onura@microsoft.com, nihan@ceng.metu.edu.tr,
ilyas@cs.bilkent.edu.tr

Abstract. As the web services proliferate and complicate it is becoming an overwhelming job to manually prepare the web service compositions which describe the communication and integration between web services. This paper analyzes the usage of the Event Calculus, which is one of the logical action-effect definition languages, for the automated preparation and execution of web service compositions. In this context, abductive planning capabilities of the Event Calculus are utilized. It is shown that composite process definitions in OWL-S can be translated into Event Calculus axioms so that planning with generic process definitions is possible within this framework.

Keywords: Event Calculus, Web Service Composition, Planning.

1 Introduction

Web services can be described as a set of related functionalities that can be programmatically accessed through the web protocols [2]. The distribution of the functions of the business through web services helped a lot to integrate services of different companies. However as the web applications flourished and the number of web services increase another difficulty appeared in the horizon. Application integrators now are concerned with finding the correct web service that meets the demands of the customer while building the applications. In such a dynamic domain, automatic integration or composition of web services would be helpful since the unknowns of the demands are too much or too diverse. This brings us to the problem of automatic web services composition.

Given a repository of service descriptions and a service request, the *web service composition problem* involves finding multiple web services that can be put together in correct order of execution to obtain the desired service. Finding a web service that can fulfill the request alone is referred to as *web service discovery problem*. When it is impossible for one web service to fully satisfy the request, one the other hand, one has to compose multiple web services, in sequential or parallel, preferably in an automated fashion.

* This work is supported by the Scientific and Technical Research Council of Turkey, METU-ISTEC Project No: EEEAG 105E068.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 142–157, 2008.

However, automated composition of services is a hard problem and it is not entirely clear which techniques serve the problem best. One family of techniques that has been proposed for this task is AI planning. The general assumption of such kind of methods is that each Web service can be specified by its preconditions and effects in the planning context. The preconditions and effects are the input and the output parameters of the service respectively. In general, if the user can specify the preconditions and effects required by the composite service, a plan or process is generated automatically by logical theorem prover or AI planners. The automation of web services composition may mean two things: either the method can generate the process model automatically, or the method can locate the correct services if an abstract process model is given [18]. In this paper we are concerned with defining an abstract process model.

Recently, a considerable amount of work has investigated the potentials and boundaries of applying AI planning techniques to derive web service processes that achieve the desired goals [7,10,12,15,17,18,24]. As mentioned in [17], the event calculus [6] is one of the convenient techniques for the automated composition of web services. In this paper we aim to demonstrate how the event calculus can be used in the solution of this problem. Our goal is to show that the event calculus can be used to define an abstract composite process model and produce a user specific composition (plan). Abductive planning of the event calculus [21] is used to show that when atomic services are available, the composition of services that would yield the desired effect is possible. The problem of matching the input/output parameters to find a web service in a given repository is out of the scope of this work. We assume that these matching tasks are pre-processed and selected.

The idea of using the event calculus in the context of web services and interactions in multiagent systems is not new [4,19,23,25]. In [4], an event calculus based planner is used in an architecture for automatic workflow generation on the Web/Grid. This work is closely related to our work, however since the details of the formalism is not given it is not possible to compare it with ours. In [19] the event calculus has been used in verifying composed web services, which are coordinated by a composition process expressed in WSBPEL. Here the aim is to verify a composed service, not generating the composition itself. The work in [23] attempts to establish a link between agent societies and semantic web-services, but uses another version of the event calculus which avoids abduction and stick to normal logic programs. In [25] an approach for formally representing and reasoning about commitments in the event calculus is developed. This approach is applied and evaluated in the context of protocols, which represent the interactions allowed among communicating agents.

In this paper our aim is to contribute the research along this direction by presenting a formal framework that shows how generic composition procedures are described in the event calculus to produce specific plans for the requested goals. Our main contribution is the translation of OWL-S to event calculus and demonstrating how planning with complex actions is done within this framework. We present the event calculus framework as an alternative approach for building agent technology, based on the notion of generic procedures and customizing user constraints.

The rest of the paper is organized as follows. Section 2 gives insight information about current technologies, the web service composition problem and techniques used to solve the problem. In Section 3, the event calculus as a logical formalism and its abduc-

tive implementation are explained. In Section 4, we present the use of abductive event calculus in the solution of automated web services composition problem. Section 5 presents a translation of OWL-S service descriptions to the event calculus and how the abductive event calculus can be used to define abstract process model needed for composition. Finally, Section 6 presents conclusions and possible future work.

2 Related Work

Building composite Web services with an automated or semi-automated tool is a critical and hard task. This problem has received a lot of attention recently [18]. In the literature, AI planning algorithms have been widely used to automatically compose web services [9,10,17,24]. Most apparent reason behind this preference is the great similarities between these two fields.

Both the planning problem and composition problem seek a (possibly partially) ordered set of operations that would lead to the goal starting from an initial state (or situation). Operations of the planning domain are actions (or events) and operations of the composition domain are the web services. Like actions, Web services have parameters, preconditions, results and effects hence they are very attractive to be used in conventional planning algorithms.

Viewing the composition problem as an AI planning problem, different planners are employed for the solution. An excellent survey of modern planning algorithms and their application to web service composition problem can be found in [17]. Here we highlight some of the existing work that is most relevant to our approach.

Estimated-regression is a planning technique in which the situation space is searched with the guide of a heuristic that makes use of backward chaining in a relaxed problem space [10]. In this approach, the composition problem is seen as a PDDL planning problem and efforts are condensed to solve the problem in PDDL domain referring to the common difficulties of Web Services domain. In fact, a translator [11] has been written which converts DAML-S (former version of OWL-S) and PDDL into each other. This shows that the composition problem can be (informally) reduced to a planning problem and in that sense working in PDDL domain is not much different indeed.

In [12], web service composition problem is assumed to be the execution of generic compositions with customizable user constraints. GOLOG [8], which is a situation calculus implementation with complex actions, is used to write the generic processs model (complex action). It is said to be generic since it is not executable without user constraints. After the user specifies the constraints, it is executed and the solver tries to generate the plan according to the runtime behavior of the services. The output of this method is a running application which satisfies the user requests.

Hierarchical Task Network (HTN) planning has been applied to the composition problem to develop software to automatically manipulate DAML-S process definitions and find a collection of atomic processes that achieve the task [24]. SHOP2, an HTN planner, is used to generate plans in the order of its execution. This work has recently been extended into another planning algorithm called Enquirer, which provides information gathering facilities during planning [7].

In [15], a taxonomy is presented for the classification of the web service composition problem. This taxonomy is used to help select the right solution for the composition problem at hand. According to this classification, the Event Calculus based approach falls in the category of AI planning methods that best suits to the solution of small scale and simple operator based automated web service compositions.

3 Event Calculus

Event calculus [6] is a general logic programming treatment of time and change. The formulation of the event calculus is defined in first order predicate logic like the situation calculus. Likewise, there are actions and effected fluents. Fluents are changing their valuations according to effect axioms defined in the theory of the problem domain. However there are also big differences between both formalisms. The most important one is that in the event calculus, narratives and fluent valuations are relative to time points instead of successive situations. The most appearing advantage of this approach is the inherent support for concurrent events. Events occurring in overlapping time intervals can be deduced. Inertia is an assumption, which accounts a solution to the frame problem together with other techniques and it is saying that a fluent preserves its valuation unless an event specified to affect (directly or indirectly) the fluent occurs.

Each event calculus theory is composed of axioms[1]. A fluent that holds since the time of the initial state can be described by the following axioms [20]:

$$holdsAt(F, T) \leftarrow initially(F) \wedge \neg clipped(t_0, F, T)$$
$$holdsAt(\neg F, T) \leftarrow initially(\neg F) \wedge \neg declipped(t_0, F, T)$$

Axioms below are used to deduce whether a fluent holds or not at a specific time.

$$holdsAt(F, T) \leftarrow$$
$$happens(E, T_1, T_2) \wedge initiates(E, F, T_1) \wedge \neg clipped(T_1, F, T) \wedge T_2 < T$$
$$holdsAt(\neg F, T) \leftarrow$$
$$happens(E, T_1, T_2) \wedge terminates(E, F, T_1) \wedge \neg declipped(T_1, F, T) \wedge T_2 < T$$

The predicate *clipped* defines a time frame for a fluent that is overlapping with the time frame of an event which terminates this fluent. Similarly *declipped* defines a time frame for a fluent which overlaps with the time frame of an event that initiates this fluent. The formula $initiates(E, F, T)$ means that fluent F holds after event E at time T. The formula $terminates(E, F, T)$ denotes that fluent F does not hold after event E at time T. The formula $happens(E, T_1, T_2)$ indicates that event E starts at time T_1 and end at time T_2. The instantaneous events are described as $happens(E, T_1, T_1)$.

$$clipped(T_1, F, T_4) \leftrightarrow (\exists E, T_2, T_3)\ [\ happens(E, T_2, T_3) \wedge$$
$$terminates(E, F, T_2) \wedge T_1 < T_3 \wedge T_2 < T_4\ ]$$
$$declipped(T_1, F, T_4) \leftrightarrow (\exists E, T_2, T_3)\ [\ happens(F, T_2, T_3) \wedge$$
$$initiates(E, F, T2)\ \wedge\ T1 < T3 \wedge T2 < T4\ ]$$

[1] Variables begin with upper-case letters, while function and predicate symbols begin with lower-case letters. All variables are universally quantified with maximum possible scope unless otherwise indicated.

3.1 Abductive Event Calculus

Abduction is logically the inverse of deduction. It is used over the event calculus axioms to obtain partially ordered sets of events. Abduction is handled by a second order abductive theorem prover (ATP) in [21]. ATP tries to solve the goal list proving the elements one by one. During the resolution, abducible predicates, i.e. < (temporal ordering) and *happens*, are stored in a residue to keep the record of the narrative. The narrative is a sequence of time-stamped events, and the residue keeping a record of the narrative is the plan.

In this paper, the predicate *ab* is used to denote the theorem prover. It takes a list of goal clauses and tries to find out a residue that contains the narrative. For each specific object level axiom of the event calculus, a meta-level *ab* solver rule is written. For example an object level axiom in the form:

$$A_H \leftarrow AB_1 \wedge AB_2 \wedge \ldots \wedge AB_N$$

is represented with the predicate *axiom* in the ATP theory and it is translated to:

$$axiom(AH, [AB_1, AB_2, \ldots, AB_N])$$

During the resolution process axiom bodies are resolved by the *ab* which populates the *abducibles* inside the residue. A simplified version of *ab* solver is as follows.

$ab([],RL,RL,NL).$
$ab([A|GL],CurrRL,RL,NL) \leftarrow abducible(A),\ NewRL = [A|CurrRL],\ consistent(NL,NewRL),\ ab(GL,NewRL,RL,NL).$
$ab([A|GL],CurrRL,RL,NL) \leftarrow axiom(A,AL),\ append(AL,GL,NewGL),\ ab(NewGL,CurrRL,RL,NL).$
$ab([not(A)|GL],CurrRL,RL,NL) \leftarrow irresolvable(A,CurrRL),\ ab(GL,CurrRL,RL,[A|NL]).$

In this definition *GL* denotes the goal list, *RL* represents the residue list, *NL* represents the residue of negated literals, *A* is the axiom head and *AL* is the axiom body. Intuitively, the predicate *abducible* checks if the axiom is abducible. If it is so, it is added to the residue. If it is an axiom then its body is inserted into the goal list to be resolved with other axioms. Negated literals are proven by negation as failure (NAF). However as the residue grows during the resolution, the negative literals, which were previously proven, might not be proven anymore. This situation may occur when negative literals were proven due to the absence of contradicting evidence; however the newly added literals might now allow the proof of the positive of literals, invalidating the previous negative conclusions. For that reason, whenever the residue is modified, previously proven negated literals should be rechecked. The predicate *irresolvable* checks whether the negated literal is resolvable with the current residue or not. The negative literal in question might also mention a non-abducible predicate. In this case it needs to be resolved with the axioms not the residue. This possibility is studied in [21]. The predicate *consistent* checks that none of the negated literals is resolvable with the current narrative residue using the predicate *irresolvable* for each negated literal.

4 Web Services Composition with Abductive Planning

The event calculus can be used for planning as it is theoretically explained in [21]. The planning problem in the event calculus is formulated in simple terms as follows: Given the domain knowledge (i.e. a conjunction of *initiates, terminates*), the Event Calculus axioms (i.e. *holdsAt, clipped, declipped*) and a goal state (e.g. *holdsAt*(*f,t*)), the abductive theorem prover generates the plan which is a conjunction of *happens* i.e. the narrative of events, and temporal ordering predicates, giving the partial ordering of events.

4.1 Web Services

In the event calculus framework, the web services are modeled as events with input and output parameters. For instance, if a web service returns the availability of a flight between two locations, its corresponding event is given in Fig.1.

```
-- web service description
<message name='GetFlight_Request'>
<part name='Origin' type='xs:string'>
<part name='Destination' type='xs:string'>
<part name='Date' type='xs:date'>
</message>
<message name='GetFlight_Response'>
<part name='FlightNum' type='xs:string'>
</message>

-- event
getFlight(Origin, Destination, Date, FlightNum)
```

Fig. 1. Web Service to Event Translation

The web service operation *GetFlight* is translated to the event *getFlight*. The inputs and outputs of the web service are translated as parameters of the event. The invocation of the web service is represented with the predicate *happens*:

```
happens(getFlight(Origin,Destination,Date,FlightNum), T1, T1) ←
        ex_getFlight(Origin, Destination, Date, FlightNum).
```

The parameters of the event are populated with help of the predicate *ex_getFlight* which is a call to the actual web service. This predicate is used as a precondition for the event and it is invoked anytime it is added to the plan. In order to resolve literals which are non-axiomatic assertions such as conditions or external calls *ab* is extended to contain the following rule:

$$ab([L|GL], CL, RL, NL) \leftarrow \neg axiom(L) \wedge L \wedge ab(GL, CL, RL, NL)$$

In this rule *L*, *GL*, *RL* and *NL* denote, respectively, the non-axiomatic literal, the goal list, the narrative residue and the negation residue. If a non-axiomatic literal is encountered then *ab* directly tries to prove the literal and if it is successful it continues with rest of the goal list.

In ATP implementation, the external call bindings like the predicate *ex_getFlight* are loaded from an external module that is written in C++ programming language. After invoking the associated service, flight number is unified with *FlightNum*, the last parameter of *getFlight* event.

Let us assume that we have the following specific axioms for a very simple travel domain.

```
axiom( happens(getFlight(Origin,Dest,Date,FlightNum), T, T),
      [ ex_getFlight(Origin,Dest,Date,FlightNum)] ).
axiom(initiates(getFlight(Origin,Dest,Date,FlightNum),
        at_location(Dest),T),
      [ holdsAt(at_location(Origin),T), Origin \== Dest ] ).
axiom(terminates(getFlight(Origin,Dest,Date,FlightNum),
        at_location(Origin),T),
      [ holds_at(at_location(Origin),T), Origin \== Dest ] ).
axiom( initially(at_location(ankara)), []).
```

There are two non-axiomatic literals, namely \== and *ex_getFlight* in the bodies of the axioms. The predicate \== checks whether two bound variables are different or not. The predicate *ex_getFlight* represents an external web service operation, and it returns the flight number for the given origin and destination cities. Thus, the parameters of the *getFlight* event are populated. The *initiates* and *terminates* axioms describe how the fluent *at_location* is affected by *getFlight* event. The *initially* axioms says that our initial location is the city *ankara*. Let us assume that, we only have the following three flights in our travel domain, and the predicate *ex_getFlight* returns these flights one by one.

getFlight(*ankara, izmir, tk101*).
getFlight(*ankara, istanbul, tk102*).
getFlight(*istanbul, izmir, tk103*).

In order to find the possible plans for the goal of being in *izmir*, the abductive theorem prover is invoked with the goal *ab*([*holdsAt*(*at_location*(*izmir*), *t*)], `[]`, *RL*, `[]`). The theorem can find the following two plans one by one.

plan1: [*happens*(*getFlight*(*ankara, izmir, tk101*), *t1, t1*), $t1 < t$]
plan2: [*happens*(*getFlight*(*ankara,istanbul,tk102*),*t2,t2*),
happens(*getFlight*(*istanbul,izmir,tk103*), *t1, t1*), $t2 < t1$, $t1 < t$]

Here each plan contains the time stamped *happens* predicates and temporal ordering between these time stamps. The time constants in the plan (*t1* and *t2*) are generated by the abductive reasoner. The abductive planner binds the given time parameter to a unique constant if the time parameter is an unbound variable.

4.2 Plan Generation

ATP returns a valid sequence of time stamped events that leads to the goal. If there are several solutions they are obtained with the help of backtracking of Prolog. Multiple solutions could be thought as different branches of a more general plan. For instance, assume that the following event sequence is generated after a successful resolution process.

$happens(E_1, T_1, T_1)$. $happens(E_2, T_2, T_2)$. $happens(E_3, T_3, T_3)$. $T_1 < T_2 < T_3$

It can be concluded that when executed consecutively, the ordered set $\{E_1, E_2, E_3\}$ generates the desired effect to reach the goal. In addition to this plan, alternative solutions could be examined. In order to do such a maneuver, the executer should have a tree like plan where proceeding with alternative paths is possible. Assume that the following totally ordered sequences of events also reach the same goal.

$\{E_1, E_5, E_4\}, \{E_1, E_2, E_4\}, \{E_6, E_7\}$

When these separate plans are combined, a graph which describes several compositions of web services, is formed (see Fig. 2). In this graph the nodes represent the events (web services) and *CS* is the start of composition (i.e. the initial state). The nodes with the label *Exclusive-Or-Split* (*XOr*) represent alternative branches among which only one could be chosen. Also several alternative paths are joined in the nodes with the label *XOr-Join. XOr-Joins* mandate that only one of the branches is active at the joining side. This graph contains all plans (i.e. composite services) generated by the planner. This graph is used to evaluate the composed services according to the non-functionality values such as cost, quality and response time, and the best plan can be chosen afterwards, which will be executed by the execution engine.

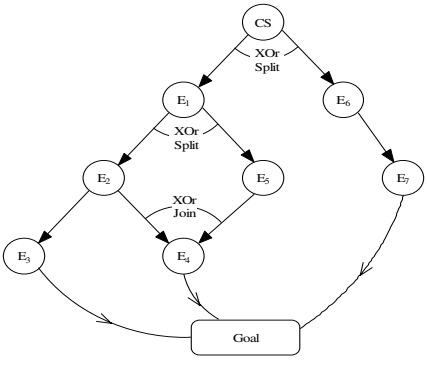


Fig. 2. All generated compositions

4.3 Concurrency of Events

The narratives generated by the ATP are partially ordered set of events. Due to the partial ordering, events, for which a relative ordering is not specified, can be thought to be concurrent. For instance, assume ATP has generated the following narrative:

$happens(E_1, T_1, T_1)$. $happens(E_2, T_2, T_2)$. $happens(E_3, T_3, T_3)$.
$happens(E_4, T_4, T_4)$. $T_1 < T_2 < T_4$, $T_1 < T_3 < T_4$

Since there is no relative ordering between E_2 and E_3 they are assumed to be concurrent. If this is the only narrative generated by the ATP then the plan can be shown as in Fig. 3.

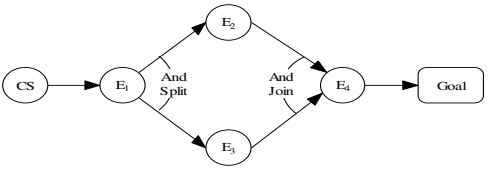


Fig. 3. Concurrent Composition

In this graph, concurrent events are depicted as *And-Split* since both of the branches should be taken after the event E_1. Before the event E_4 *And-Join* is required since both of E_2 and E_3 should be executed.

5 Web Services Composition with Generic Process Definition

In the literature, one of the most promising leaps on automating the Web Service Composition is taken with the OWL-S language [16]. In OWL-S, Web Services are abstracted, composed and bound to concrete service providers. Web Services are composed by a series of operations, which atomically provide certain functions. Service interactions can be as simple as a single operation invocation (e.g. http://www.random.org returns random numbers with a single Web Service operation). They can be as complicated as a multi-department electronic commerce site for shopping, where catalog browsing, item selection, shipment arrangements and payment selection are accomplished by invoking a series of operations. (e.g. Amazon Web Service http://www.amazon.com).

Several atomic processes constitute a *Composite Process w*hen connected with the flow control constructs of OWL-S. If an automated system requires the provided service it should execute the composite processes as they are defined in the OWL-S, supplying the intermediate inputs to the atomic services nested under them.

The Event Calculus framework can be used to define composite processes (i.e. complex goals) and ATP can be used to generate a plan which corresponds to the user specific composition of the web service. Composite processes will correspond to compound events in the Event Calculus [3]. Like the composite processes, compound events provide the grouping of sub-events. In the following sections, first, an OWL-S to event calculus translation scheme is presented to show that OWL-S composition constructs can be expressed as event calculus axioms[2]. Then an example application will be presented to illustrate the use of generic process definition and its use in the abductive event calculus planner.

5.1 OWL-S to Event Calculus Translation

Composite processes are composed of control constructs which closely resemble to standard workflow constructs. Since further composite processes can be used inside a

[2] For readability purposes, we will omit axiom predicate in the rest of the paper and present object level axioms only. However, note that they are converted into the axiom predicate in the implementation.

composite process, the translation is recursively applied until all composite processes are replaced with the corresponding axioms that contain atomic processes. Here we present an OWL-S to event calculus translation scheme. The automatic mapping is possible, but we have not implemented it yet.

5.1.1 Atomic Processes

Atomic processes are translated into simple events of the Event Calculus. An abstract representation of an atomic process of OWL-S is given in Fig. 4.

```
Atomic Process<A, V, P, E, O, E^C, O^C>
  A : Atomic Process Functor
  V : Set of Inputs: {V_1, V_2, ..., V_N}
  P : Preconditions: Conjunction of Literals (P_1 ∧ P_2 ∧ ... ∧ P_M)
  E : Effects: Conjunction of Literals (E_1 ∧ E_2 ∧ ... ∧ E_K)
  O : Outputs: Set of Outputs {O_1, O_2, ..., O_L}
  E^C : Conditional Effects: Set of literals {E^C_1, E^C_2, ..., E^C_R}
        where each E^C_i has a condition such as
        E^C_i ← BE^C_i : BE^C_i are conjunction of literals
  O^C : Conditional outputs
```

Fig. 4. Atomic Process Definition of OWL-S

This definition is translated to the Event Calculus as an event with the same name as the atomic process *A* and the effect axioms are defined according to the preconditions and effects. The translation is given in Fig. 5.

```
initiates(A(V, O), E_i, T) ← holdsAllAt(P, T) ∧ invoke(A, V, O, T)
        where E_i ∈ E^+ (positive literals of E)
terminates(A(V, O), E_i, T) ← holdsAllAt(P, T) ∧ invoke(A, V, O, T)
        where E_i ∈ E^- (negative literals of E)
initiates(A(V, O), E^C_i, T) ← holdsAllAt(P, T) ∧
                holdsAllAt(BE^C_i, T) ∧ invoke(A, V, O, T)
        where E^C_i ∈ E^C+
terminates(A(V, O), E^C_i, T) ← holdsAllAt(P, T) ∧
                holdsAllAt(BE^C_i, T) ∧ invoke(A, V, O, T)
        where E^C_i ∈ E^C-
holdsAllAt({F_1, F_2, ..., F_Z}, T) ↔ holdsAt(F_1, T) ∧
                holdsAt(F_2, T) ∧ ... ∧ holdsAt(F_Z, T)
```

Fig. 5. Atomic Process Translation

The meta predicate *holdsAllAt* has an equivalent effect of conjunction of *holdsAt* for each fluent that *holdsAllAt* covers. The predicate *invoke* is used in the body of effect axioms to generate the desired outputs (it corresponds to the invocation of external calls through the happens clause as illustrated in the example in Section 4.1). It takes the name of the atomic process, input parameters and unifies the outputs with the results of the corresponding Web Service operation invocation.

5.1.2 Composite Process Translation

Composite processes combine a set of processes (either atomic or composite) with different control constructs. An example composition which is composed of nested structures

is given in Fig. 6. Split, Join and Repeat-While control constructs are used in this composite process. It is necessary to be able to express such control constructs in the event calculus framework. This problem has been studied earlier in different contexts [3,5]. For the purpose of the web services composition problem, OWL-S constructs should be translated into compound events in the event calculus framework.

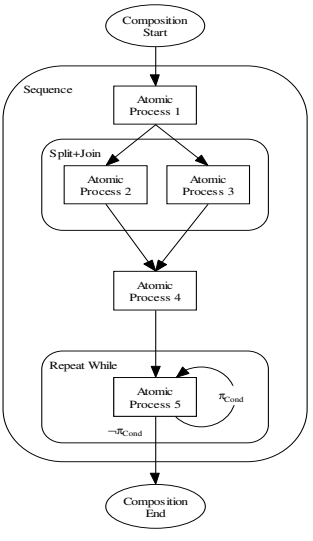


Fig. 6. Example of a Composite Process

The translation of some of the flow control constructs into the Event Calculus axioms is summarized in the following. Others can be found in [1]. Constructs are originally defined in XML (to be more precise in RDF) document structure however since they are space consuming only their abstract equivalents will be given.

Sequence

The *Sequence* construct contains the set of all component processes to be executed in order. The abstract OWL-S definition of the composite process containing a *Sequence* control construct and its translation into an Event Calculus axiom is given in Fig 7. The translation is accomplished through the use of compound events in the Event Calculus which contains sub-events. The sequence of events are triggered from the body of the compound event and the ordering between them is ensured with the predicate < (precedes).

Sequence Composite Process<C, V, P, S> *C : Composite Process Functor* *V : Set of Inputs* $\{V_1, V_2, \ldots, V_N\}$ *P : Preconditions* $(P_1 \wedge P_2 \wedge \ldots \wedge P_M)$ *S : Sequence of Sub-Processes* *Ordered set of* $\{S_1, S_2, \ldots, S_K\}$	$happens(C, T_1, T_N) \leftarrow$ $holdsAllAt(P, T_1) \wedge$ $happens(S_1, T_2, T_3) \wedge$ $happens(S_2, T_4, T_5) \wedge \ldots \wedge$ $happens(S_K, T_{2K}, T_{2K+1}) \wedge$ $T_1 < T_2 \wedge T_3 < T_4 \wedge \ldots \wedge$ $T_{2K-1} < T_{2K} \wedge T_{2K+1} < T_N$

Fig. 7. Sequence Composite Process

If-Then-Else

If-Then-Else construct contains two component processes and a condition. Its structure and translation are given in Fig. 8. Two *happens* axioms are written for both cases. With the help of *notholdsAllAt* which is logically the negation of *holdsAllAt*, the second axiom is executed when the *else-case* holds.

If-Then-Else Composite Process *<C, V, P, π, S_π, $S_{\neg\pi}$>* *C : Composite Process Functor* *V : Set of Inputs {V_1, V_2, ..., V_N}* *P : Preconditions ($P_1 \wedge P_2 \wedge ... \wedge P_M$)* *$\pi$: If condition ($\pi_1 \wedge \pi_2 \wedge ... \wedge \pi_K$)* *$S_\pi$: If condition Sub-Process* *$S_{\neg\pi}$: Else condition Sub-Process*	*happens(C, T_1, T_N) ←* *holdsAllAt(P, T_1) ∧* *holdsAllAt(π, T_1) ∧* *happens(S_π, T_2, T_3) ∧* *$T_1 < T_2 \wedge T_3 < T_N$* *happens(C, T_1, T_N) ←* *holdsAllAt(P, T_1) ∧* *notholdsAllAt(π, T_1) ∧* *happens($S_{\neg\pi}$, T_2, T_3) ∧* *$T_1 < T_2 \wedge T_3 < T_N$* *notholdsAllAt({$F_1$,$F_2$,...,$F_P$},T)* *↔ holdsAt($\neg F_1$, T) ∨* *holdsAt($\neg F_2$, T) ∨ ... ∨* *holdsAt($\neg F_N$, T)*

Fig. 8. If-Then-Else Composite Process

Repeat-While and Repeat-Until

Repeat-While and *Repeat-Until* constructs contain one component process and a loop controlling condition. The loop iterates as long as the condition holds for *Repeat-While* and does not hold for *Repeat-Until*. They have a common structure and it is given in Fig. 9. The figure presents the translation of *Repeat-While* only, since the translation of the other is similar. Two *happens_loop* axioms are written for both states of the loop condition. The composite event is triggered when preconditions hold. The body of the loop is recursively triggered as long as the loop condition permits. The preconditions and the loop condition are checked at time T_1. If they hold, the component process is invoked at a later time T_2.

Repeat-While/Unless Composite Process <C, V, P, π, S_π> *C : Composite Process Functor* *V : Set of Inputs {V_1, V_2, ..., V_N}* *P : Preconditions ($P_1 \wedge P_2 \wedge ... \wedge P_M$)* *$\pi$: Loop condition ($\pi_1 \wedge \pi_2 \wedge ... \wedge \pi_K$)* *$S_\pi$: Loop Sub-Process*	*happens(C, T_1, T_N) ←* *holdsAllAt(P, T_1) ∧* *happens_loop(C, π, T_1, T_N).* *happens_loop(C, π, T_1, T_N) ←* *holdsAllAt(π, T_1) ∧* *happens(S_π, T_2, T_3) ∧* *happens_loop(C, π, T_4, T_5) ∧* *$T_1 < T_2 \wedge T_3 < T_4 \wedge T_5 < T_N$* *happens_loop(C, π, T_1, T_1) ←* *notholdsAllAt(π, T1)*

Fig. 9. Repeat-While/Unless Composite Process

In the given abstract translations, it may seem that the set of inputs are not used, but the actual translations spread out the contents of the set of inputs as the appropriate parameters of the component processes. This is illustrated in the example given in Section 5.2.

5.2 Example of a Composition

In this section we illustrate the use of the abductive event calculus in generating compositions from a given composite procedure. The example illustrates how one can describe a complex goal and find a plan to achieve that goal.

The implementation of the traveling problem given in [12] is formulated in the Event Calculus. In [12], a generic composition is presented for the traveling arrangement task. In this procedure, the transportation and hotel booking are arranged and then mail is sent to the customer. Finally an online expense claim is updated. The transportation via air is selected with the constraint that it should be below the customer's specified maximum price. If the destination is close enough to drive by car then instead of air transportation, car rental is preferred. The customer specifies a maximum drive time for this purpose. If the air transportation is selected then a car is arranged for local transportation. Also a hotel is booked for residence at the destination.

Compound events are used to express generic compositions in the Event Calculus in a similar way that they have been used in OWL-S translation. The whole operation is decomposed into smaller tasks which are separately captured with other compound events [1]. The Event Calculus translation is given in Fig. 10.

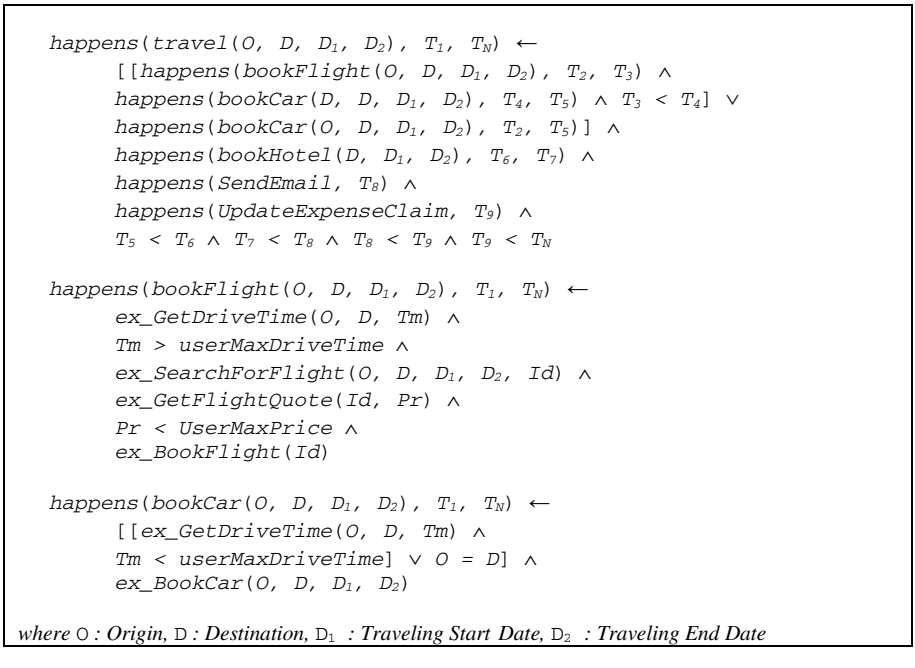


Fig. 10. Generic Composition in the Event Calculus

In this translation *userMaxDriveTime* and *userMaxPrice* are the user preference values which alter the flow of operations. Based on traveling inputs and user preferences the traveling arrangement is accomplished with the help of external Web Service calls (in Fig. 10 they are represented with predicates with *ex_* prefix). When this composition is implemented in the ATP, a residue which contains the sequence of events to arrange a travel will be returned as the plan. For instance let us assume that we have the definitions of several external web services for the atomic processes like *GetDriveTime, SearchForFlight, GetFlightQuote etc.*), and we have the following initiates axiom:

initiates(*travel*(*O,D,SDate,EDate*), *travelPlanned*(*O,D,SDate,EDate*),*T*).

If we want to find a travel plan from Ankara to Athens between the dates October 22 and October 24, we can invoke the ATP with the following goal:

ab([*holdsAt*(*travelPlanned*(*ankara,athens,october22,october24*), *t*)], *R*)

The variable *R* will be bound to a plan, for instance, of the following form:

```
[ hapens(updateExpenseClaim, t7, t7),
  happens(sendEmail, t6, t6),
  happens(bookHotel(athens, october22, october24), t5, t5),
  happens(bookCar(athens, athens, october22, october24), t4, t4),
  happens(bookFlight(ankara, athens, october22), t3, t3),
  happens(travel(ankara, athens, october22, october24), t1, t2)
  t7 < t2,  t6 < t7,  t5  < t6,  t4 < t5,  t3 < t4,  t1 < t3,  t2 < t ].
```

The plan shows which web services must be invoked for the composition and also the temporal ordering among them.

6 Conclusions

In this paper, the use of the event calculus has been proposed for the solution of web service composition problem. It is shown that when a goal situation is given, the event calculus can find proper plans as web service compositions with the use of abduction technique. It is also shown that if more than one plan is generated, the solutions can be compiled into a graph so that the best plan can be chosen by the execution engine.

In [24], SHOP2 is used to translate DAML-S process model into SHOP2 operators. This translation assumes certain constraints for the process model to be converted. The first assumption in SHOP2 translation is that the atomic processes are assumed to be either output generating or effect generating but not both. Atomic processes with conditional effects and outputs are not converted at all. Our translation supports atomic processes with outputs, effects and conditional effects. Another limitation of SHOP2 translation is the support for concurrent processes. Since SHOP2 cannot handle parallelism the composite constructs *Split* and *Split+Join* cannot be translated. On the other hand, our translation supports for these constructs since event calculus is inherently capable of handling concurrency.

Both the event calculus and GOLOG can be used to express composite process models. The most important difference between the Event Calculus and GOLOG is the syntax of the languages. GOLOG provides extra-logical constructs which ease the definition of the problem space as it is given in [12] for the same example above.

These constructs can be easily covered with Event Calculus axioms too. Furthermore, since the event calculus supports time points explicitly, it is easier to model concurrency and temporal ordering between actions in the Event Calculus. Therefore it is more suitable to the nature of web services composition problem with respect to expressiveness and ease of use.

As a future work, the results that are theoretically expressed in this paper will be put into action and implemented within a system which works in a real web environment. It would be helpful if a language is developed for the event calculus framework, in order to define common control structures of web service compositions in a more direct way. In fact, there has been already some efforts along this direction, i.e. extending the event calculus with the notions of processes [3,5].

Evaluation and execution of the generated plans are the final phases of automatic web service composition. These phases are left out of the scope of this paper. However, in a realistic implementation, these issues and other aspects like normative ones need to be studied. It would be interesting to formalise the rights, responsibilities, liabilities that are created by composing different web services [9].

As another further work, it is worth trying event calculus planners that employ SAT solvers for efficiency reasons.

References

1. Aydin, O.: Automated web service composition with the event calculus, M.S. Thesis, Dept. of Computer Engineering, METU, Ankara (2005)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American Magazine (May 2001)
3. Cicekli, N.K., Cicekli, I.: Formalizing the specification and execution of workflows using the event calculus. Information Sciences (to appear)
4. Chen, L., Yang, X.: Applying AI Planning to Semantic Web Services for workflow Generation. In: Proc. of the 1st Intl. Conf. on Semantics, Knowledge and Grid (SKG 2005) (2005)
5. Jacinto, J.D.: REACTIVE PASCAL and the event calculus: A platform to program reactive, rational agents. In: Proc. of the Workshop at FAPR 1995: Reasoning about Actions and Planning in Complex Environments (1996)
6. Kowalski, R.A., Sergot, M.J.: A Logic-Based Calculus of Events. New Generation Computing 4(1), 67–95 (1986)
7. Kuter, U., Sirin, E., Nau, D., Parsia, B., Hendler, J.: Information gathering during planning for web service composition. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298. Springer, Heidelberg (2004)
8. Levesque, H., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.: GOLOG: A Logic programming language for dynamic domains. Journal of Logic Programming 31(1-3), 59–84 (1997)
9. Marjanovic, O.: Managing the normative context of composite e-services. In: ICWS-Europe, pp. 24–36 (2003)
10. McDermott, D.: Estimated-regression planning for interactions with Web Services. In: Sixth International Conference on AI Planning and Scheduling. AAAI Press, Menlo Park (2002)

11. McDermott, D.V., Dou, D., Qi, P.: PDDAML, An Automatic Translator Between PDDL and DAML, http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator1.html
12. McIlraith, S.A., Son, T.: Adapting Golog for composition of semantic Web services. In: Proceedings of Eight International Conference on Principles of Knowledge Representation and Reasoning, pp. 482–493 (2002)
13. McIlraith, S.A., Son, T., Zeng, H.: Semantic Web services. IEEE Intelligent Systems, March/April (2001)
14. Medjahed, B., Bouguettaya, A., Elmagarmid, A.K.: Composing web services on the semantic web. The VLDB Journal 12(4), 333–351 (2003)
15. Oh, S.G., Lee, D., Kumara, S.R.T.: A comparative Illustration of AI planning-based web services composition. ACM SIGecom Exchanges 5, 1–10 (2005)
16. OWL-S: Semantic Markup for Web Services Version 1.1, November 2004. Publish of Semantics Web Services Language (SWSL) Committee (Last Accessed: 17 September 2005), http://www.daml.org/services/owl-s/1.1/overview/
17. Peer, J.: Web Service Composition as AI Planning- a Survey*, Technical report, Univ. of St. Gallen, Switzerland (2005), http://elektra.mcm.unisg.ch/pbwsc/docs/pfwsc.pdf
18. Rao, J., Su, X.: A Survey of Automated Web Service Composition Methods. In: Proceedings of First International Workshop on Semantic Web Services and Web Process Composition (July 2004)
19. Rouached, M., Perrin, O., Godart, C.: Towards formal verification of web service composition. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102. Springer, Heidelberg (2006)
20. Shanahan, M.P.: The Event Calculus Explained. In: Veloso, M.M., Wooldridge, M.J. (eds.) Artificial Intelligence Today. LNCS (LNAI), vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
21. Shanahan, M.P.: An abductive event calculus planner. Journal of Logic Programming 44(1-3), 207–240 (2000)
22. Sirin, E., Hendler, J., Parsia, B.: Semi-automatic Composition of Web Services using Semantic Descriptions. In: Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS 2003 (2002)
23. Stathis, K., Lekeas, G., Kloukinas, C.: Competence checking for the global e-service society using games. In: O'Hare, G., O'Grady, M., Dikinelli, O., Ricci, A. (eds.) Proceedings of Engineering Societies in the Agents World (ESAW 2006) (2006)
24. Wu, D., Sirin, E., Parsia, B., Hendler, J., Nau, D.: Automatic web services composition using SHOP2. In: Proceedings of Planning for Web Services Workshop, ICAPS 2003 (June 2003)
25. Yolum, P., Singh, M.: Reasoning About Commitments in the Event Calculus: An Approach for Specifying and Executing Protocols. Annals of Mathematics and AI 42(1-3) (2004)

OPERAS: A Framework for the Formal Modelling of Multi-Agent Systems and Its Application to Swarm-Based Systems

Ioanna Stamatopoulou[1], Petros Kefalas[2], and Marian Gheorghe[3]

[1] South-East European Research Centre, Thessaloniki, Greece
istamatopoulou@seerc.org
[2] Department of Computer Science, CITY College, Thessaloniki, Greece
kefalas@city.academic.gr
[3] Department of Computer Science, University of Sheffield, UK
M.Gheorghe@dcs.shef.ac.uk

Abstract. Swarm-based systems are a class of multi-agent systems (MAS) of particular interest because they exhibit emergent behaviour through self-organisation. They are biology-inspired but find themselves applicable to a wide range of domains, with some of them characterised as mission critical. It is therefore implied that the use of a formal framework and methods would facilitate modelling of a MAS in such a way that the final product is fully tested and safety properties are verified. One way to achieve this is by defining a new formalism to specify MAS, something which could precisely fit the purpose but requires significant period to formally prove the validation power of the method. The alternative is to use existing formal methods thus exploiting their legacy. In this paper, we follow the latter approach. We present *OPERAS*, an open framework that facilitates formal modelling of MAS through employing existing formal methods. We describe how a particular instance of this framework, namely $OPERAS_{XC}$, could integrate the most prominent characteristics of finite state machines and biological computation systems, such as X-machines and P Systems respectively. We demonstrate how the resulting method can be used to formally model a swarm system and discuss the flexibility and advantages of this approach.

1 Introduction

Despite the counter arguments which justifiably raise concerns about formal methods, there is still a strong belief by the academic community that the development of mission critical systems demands the use of such methods for modelling, verification and testing. Opposition puts forward a significant drawback; the more complex a system is, the more difficult the modelling process turns out to be and, in consequence, the less easy it is to ensure correctness at the modelling and implementation level. Correctness implies that all desired safety properties are verified at the end of the modelling phase and that an appropriate testing technique is applied to prove that the implementation has been built

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 158–174, 2008.

in accordance to the verified model. The formal methods community has made significant progress towards the development of correct systems.

On the other hand, multi-agent systems (MAS) are complex software systems by default. Especially when the agent society grows, interaction and communication increase within a complex structure which involves variety of knowledge, abilities, roles and tasks. Nature seems to have found ways to deal with complex structures quite effectively. Consider, for example, biological systems from the smallest living elements, the cells, and how they form tissues in organisms to entire ecosystems and how they evolve [1]. There is growing interest in investigating ways of specifying such systems. The intention is to create software that mimics the behaviour of their biological counterparts. Examples of biological systems of interest also include swarm-based systems, such as social insect colonies.

The promising feature is that these systems can be directly mapped to MAS by considering each entity as an agent, with its own behavioural rules, knowledge, decision making mechanisms and means of communication with the other entities and with the environment. The overall system's behaviour is merely the result of the agents' individual actions, the interactions among them and between them and the environment. This also points to the issue of self-organisation and how collective behavioural patterns emerge as a consequence of individuals' local interactions in the lack of knowledge of the entire environment or global control.

An additional key modelling aspect of swarm-based systems is their dynamic nature and how their structure is constantly mutated. By *structure* we imply: the changing number of agents, and either their physical placement in the environment or, more generally, the structure that is dictated by the communication channels among them. Other classes of MAS also exhibit reorganisational change, characterised as behavioural or structural change [2].

Existing wide-spread formal methods fail to provide the appropriate features in order to model such dynamic system organisation —most of them assume a fixed, static structure that is not realistic (e.g. cellular and communicating automata), since communication between two agents may need to be established or ceased at any point and also new agents may appear in the system while existing ones may be removed. It is fairly recently that the issue of structural change is attempted to be in essence dealt with, and this poses a kind of dilemma: should a completely new formal notation be devised or should existing ones be used and possibly be improved? Both approaches have complementary advantages; a new formal method will directly tackle the problem of modelling of change but existing ones will carry the legacy of formal testing and verification.

In this paper, we deal with the latter approach. The next section introduces the $OPERAS$ formal definition as a framework for modelling MAS, while Section 3 presents an instance of this framework, namely $OPERAS_{XC}$, which utilises existing formal methods. A brief description of a representative case study dealing with a swarm-based system follows in Section 4 which also deals with the formal model for the case problem in question. Finally, Section 5 discusses issues arising from our attempt and concludes the paper.

2 OPERAS: Formal Modelling of MAS

2.1 Background and Related Work

In an attempt to formally model each individual agent as well as the dynamic behaviour of the overall system, a formal method should be capable of rigorously describing all the essential aspects, i.e. knowledge, behaviour, communication and dynamics. There is a number of trade-offs on the use of formal methods for MAS. To name a few: (a) the level of abstraction should be appropriate enough to lead toward the implementation of a MAS but also be appropriate enough to mathematically express specifications that can lead to formal verification and complete testing, (b) there should be accompanying toolkits which make their adoption wider by researchers and industry but at the same time the tools provided should not deviate from the theoretical framework, (c) they ought to provide means to efficiently define complex knowledge but also be able to describe control over individual agent as well as MAS states, (d) they need to be able to easily model individual agents but also to focus on the concurrency and communication among them.

In agent-oriented software engineering, several approaches using formal methods have been proposed, each one focusing on different aspects of MAS development. For example, with respect to the issue of organisation, there is a large number of approaches employing formal methods in modelling MAS and focusing on organisational reconfiguration [3, 4, 5], specificational adaptation at run time [6] and formal methodologies to engineer organisation-based MAS [7]. Other efforts have been directed toward moving to the implementation of a MAS through refinement of the specification and developing proof theories for the architecture [8], capturing the dynamics of an agent system [9], putting emphasis on capturing and controlling the system dynamics and acting behaviour of MAS [10]. Other approaches formally specify MAS and then directly execute the specification while verifying important temporal properties [11] or guide through a prototyping process [12]. Less formal approaches, which accommodate the distinctive requirements of agents, have been proposed [13]. Additionally, there is a set of general principles for capturing the organisational structure of MAS [14] which are however linked more to implementation [15] rather than formal modelling.

On the other hand, from a purely software engineering view, a plethora of formal methods are provided (Z, VDM, FSM, Petri-Nets, CCS, CSP), with none of them alone satisfying all the above mentioned criteria for MAS, but with a rich legacy on specification, semantics, testing and verification. Other formal methods, such as π-calculus, mobile ambients and P Systems with mobile membranes [16, 17, 18, 19], successfully deal with the dynamic nature of systems and concurrency of processes but lack intuitiveness when it comes to the modelling of an individual agent (lack of primitives and more complex data structures). Lately, new computation approaches as well as programming paradigms inspired by biological processes in living cells, introduce concurrency as well as neatly tackle the dynamic structure of multi-component systems (P Systems, Brane Calculus,

Gamma, Cham, MGS) [20, 21, 22]. An interesting comparison of various formal methods for the verification of emergent behaviours in swarm-based systems is reported in [23], where an asteroid exploration scenario by autonomous spacecrafts is considered. We will use the same scenario in order to benchmark our approach.

2.2 OPERAS Definition

Our aim is to define a framework in which we can use existing formal methods to model classes of MAS where self-organisation and emergent behaviour is achieved through a number of changes in their structure. As said in the previous section, none of the existing formal methods qualify to deal equally well with individual agent modelling as well as dynamics of the system. We believe that the problem will be solved by combining formal methods. But for doing so, we should somehow distinguish between the modelling of the individual agents (behaviour) and the rules that govern the change in the structure of the collective MAS (structure mutation). This distinction, which would greatly assist the modeller by breaking down the work into two separate and independent activities, may be achieved by considering that each agent is wrapped by a separate mechanism: a structural mutator. Extending an agent with a kind of a wrapper is not a novel idea in MAS engineering though it has been primarily used for communication purposes and not in the context of formal specification. In this case, we refer to a structural mutator as the independent part of the agent that is responsible for checking an agent's internal computation state and its local environment in order to determine whether a structural change in the system has to take place, might that be the addition/removal of communication channels or other agents.

In general terms, when modelling a MAS, one should specify a number of agents, the environment in which they operate, the stimuli provided from the environment as percepts to the agents, the agents abilities and roles, the agents grouping and organisation and communication between them. A *Multi-Agent System* model in its general form, as it is perceived from a formal modelling perspective can be defined by the tuple (O, P, E, R, A, S) containing:

- a set of reconfiguration rules, O, that define how the system structure evolves by applying appropriate reconfiguration operators;
- a set of percepts, P, for the agents;
- the environment's model / initial configuration, E;
- a relation, R, that defines the existing communication channels;
- a set of participating agents, A, and
- a set of definitions of types of agents, S, that may be present in the system.

The definition is general enough not to restrict any organisational structure that might be considered for the implementation of a MAS. In addition, the definition could be further extended to include protocols or other features of MAS that a modeller would wish to formally specify. For now, *OPERAS* fits our purpose, that of modelling swarm-based systems. More particularly:

- the rules in O are of the form *condition* $\Rightarrow$ *action* where *condition* refers to the computational state of agents and *action* involves the application of one or more of the operators that create / remove a communication channel between agents or introduce / remove an agent into / from the system;
- P is the distributed union of the sets of percepts of all participating agents;
- $R : A \times A$ with $(A_i, A_j) \in R$, $A_i, A_j \in A$ meaning that agent A_i may send messages to agent A_j;
- $A = \{A_1, \ldots A_n\}$ where A_i is a particular agent defined in terms of its individual behaviour and its local mechanism for structure mutation;
- $S_k = (Behaviour_k, StructureMutator_k) \in S$, $k \in Types$ where $Types$ is the set of identifiers of the types of agents, $Behaviour_k$ is the part of the agent that deals with its individual behaviour and $StructureMutator_k$ is the local mechanism for structure reconfiguration; each participating agent A_i of type k in A is a particular instance of a type of agent: $A_i = (Beh_k, StrMut_k)_i$.

2.3 OPERAS as an Open Framework

The general underlying idea is that an agent formal model consists of two parts, its behaviour and its structural mutator. The behaviour of an agent can be modelled by a formal method with its computation being driven by percepts from the environment. The structural mutator can be modelled by a set of reconfiguration rules which given the computation states of agents can change the structure of the system. The MAS structure is determined through the relation that defines the communication between the agents. The set of participating agents are instances of agent types that may participate in the system. This deals with the fact that an agent may be present at one instance of the system but disappear at another or that a new agent or a new role comes into play during the evolution of the MAS. This assumes that all agent types and roles that may participate in the system should be known in advance.

There are still some open issues which, however, make the *OPERAS* approach a framework rather than a formal method. These are: (i) Which formal method may we use in order to model the agents' behaviour? (ii) Which formal method may we use in order to model the structural mutator? (iii) Could the methods in (i) and (ii) be different? (iv) Should the formal method used in (i), for modelling behaviour, provide features for communication directly or indirectly (implicitly through percepts from the environment) among agents' behaviours? (v) Should the formal method used in (ii), for modelling structure mutation, provide features for communication directly or indirectly (through the environment)among agents' structure mutators? (vi) Which method chosen from (i) or from (ii) drives the computation of the resulting system? There is no unique answer to these questions but the choice of formal methods which are considered suitable to model either behaviour or structure mutation may affect the final model developed.

It is therefore implied that there are several options which could instantiate *OPERAS* into concrete modelling methods. Regarding the modelling of each

type of agent S_k, there are more than one options to choose from in order to specify its behavioural part and the same applies for its structure mutation mechanism. We have long experimented with two formal methods, which are X-machines with its communicating counterpart and Population P Systems (PPS) with active cells. We use X-machines because they demonstrated considerable power in modelling reactive systems and most importantly they are accompanied by two distinctive features: a complete testing strategy and a well-defined model checking methodology. We chose Population P systems because of their theoretically sound way to model computation taking place inside a membrane-like dynamic system. Ad hoc integration of these two methods [24, 25, 26] gave us some preliminary results which led us to the current combined approach we take for *OPERAS*. It is interesting to notice that none of the two formal methods by itself could successfully (or at least intuitively) model a MAS [24, 25]. This is also true, although with better results, if we use only PPSs under the OPERAS framework ($OPERAS_{CC}$) [27]. The problem still exists for other formal methods too, which means the current framework gives the opportunity to combine those methods that may be best suited to either of the two modelling tasks. In the following, we present an instance of *OPERAS*, named $OPERAS_{XC}$, that uses Communicating X-machines and features from PPSs.

3 OPERAS$_{XC}$

3.1 Modelling Behaviour

X-machines (XM), a state-based formal method introduced by Eilenberg [28], are considered suitable for the formal specification of a system's components. Stream X-machines, in particular, were found to be well-suited for the modelling of reactive systems. Since then, valuable findings using the X-machines as a formal notation for specification, communication, verification and testing purposes have been reported [29, 30, 31]. An X-machine model consists of a number of states and also has a memory, which accommodates mathematically defined data structures. The transitions between states are labelled by functions. More formally, a stream X-machine is defined as the 8-tuple $(\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$ where:

- Σ and Γ are the input and output alphabets respectively;
- Q is the finite set of states;
- M is the (possibly) infinite set called memory;
- Φ is a set of partial functions φ that map an input and a memory state to an output and a possibly different memory state, $\varphi : \Sigma \times M \rightarrow \Gamma \times M$;
- F is the next state partial function, $F : Q \times \Phi \rightarrow Q$, which given a state and a function from the type Φ determines the next state. F is often described as a state transition diagram;
- q_0 and m_0 are the initial state and initial memory respectively.

X-machines can be thought to apply in similar cases where StateCharts and other similar notations do. In principle, X-machines are considered a generalisation of models written in such formalisms.

In addition to having stand-alone X-Machine models, communication is feasible by redirecting the output of one machine's function to become input to a function of another machine. The structure of a *Communicating X-machines* (CXM) system is defined as the graph whose nodes are the components and edges are the communication channels among them (Fig. 1). A formal definition of CXMs can be found in [24].

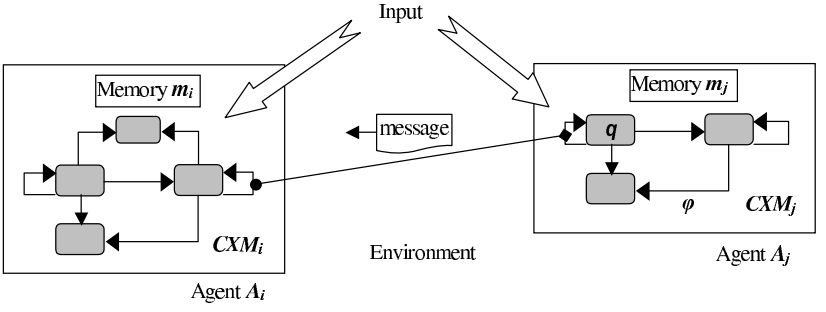


Fig. 1. An abstract system consisting of two CXM components. Communication is established by redirecting the output of a function (⧫ symbol) to another machine's function which takes it as input (• symbol).

CXMs provide a straightforward way for dealing with an agent's behaviour, however, the structure of a communicating system must be known beforehand and fixed throughout the computation.

3.2 Modelling Structure Mutation

A *Population P System* [32] is a collection of different types of cells evolving according to specific rules and capable of exchanging biological / chemical substances with their neighbouring cells (Fig. 2). More formally, a PPS is defined as a construct $\mathcal{P} = (V, K, \gamma, \alpha, w_E, C_1, C_2, \ldots, C_n, R)$ where:

- V is a finite alphabet of symbols called objects;
- K is a finite alphabet of symbols, which define different types of cells;
- $\gamma = (\{1, 2, \ldots n\}, A)$, with $A \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$, is a finite undirected graph;
- α is a finite set of bond-making rules;
- $w_E \in V^*$ is a finite multi-set of objects initially assigned to the environment;
- $C_i = (w_i, t_i)$, for each $1 \leq i \leq n$, with $w_i \in V^*$ a finite multi-set of objects, and $t_i \in K$ the type of cell i;
- R is a finite set of rules dealing with object transformation, object communication, cell differentiation, cell division and cell death.

Transformation rules replace an object within a cell. *Communication rules* allow the exchange of objects between neighbouring cells, or a cell and the environment, according to the cell type and the existing bonds among the cells. *Cell*

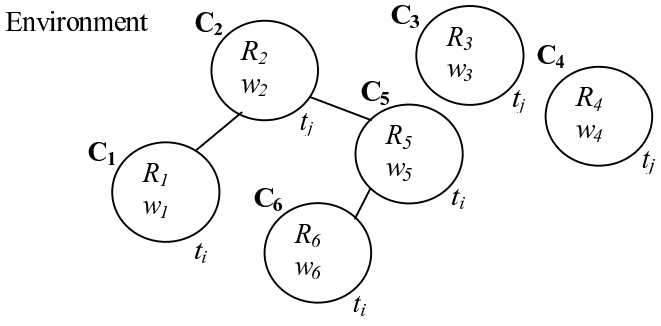


Fig. 2. An abstract example of a Population P System; C_i: cells, R_i: sets of rules related to cells; w_i: multi-sets of objects associated to the cells.

differentiation rules change a cell, transforming it into a cell of a new type. *Cell division rules* divide a cell into two cells. *Cell death rules* cause the removal of a cell from the system.

At each computation cycle, all rules regarding the transformation and communication of objects that may be applied in a cell are applied. Additionally, one out of the applicable cell differentiation, division or death rules, non-deterministically chosen, is also applied in each cell. When computation in all cells has finished, the graph is decomposed and restructured according to the specified bond-making rules in α that define the conditions under which two cells are able to communicate.

PPS provide a straightforward way for dealing with the change of a system's structure, however, the rules specifying the behaviour of the individual cells (agents) are more commonly of the simple form of rewrite rules which are not sufficient for describing the behaviour of the respective agent.

3.3 Definition of OPERAS$_{XC}$

We may now move on to a more formal $OPERAS_{XC}$ definition that uses both a CXM (indicator subscript X) and PPS-cell-inspired construct (indicator subscript C) for specifying each of the agents. An abstract example of an $OPERAS_{XC}$ model consisting of two agents is depicted in Fig. 3.

For the following, we consider that the computation state of a CXM describing the behaviour of an agent is a 3-tuple $Q \times M \times \Phi$ that represents the state the XM is in (q_i), its current memory (m_i) and the last function that has been applied (φ_i).

A MAS in $OPERAS_{XC}$ is defined as the tuple (O, P, E, R, A, S) where:

- The rules in O are of the form $condition \Rightarrow action$ where $condition$ is a conjunction of (q, m, φ) and $action$ involves the application of one or more of the operators attachment **ATT** and detachment **DET**, which reconfigure the communication channels among existing CXMs and generation **GEN** and destruction **DES**, which generate or destroy an agent in/from the system. Additional

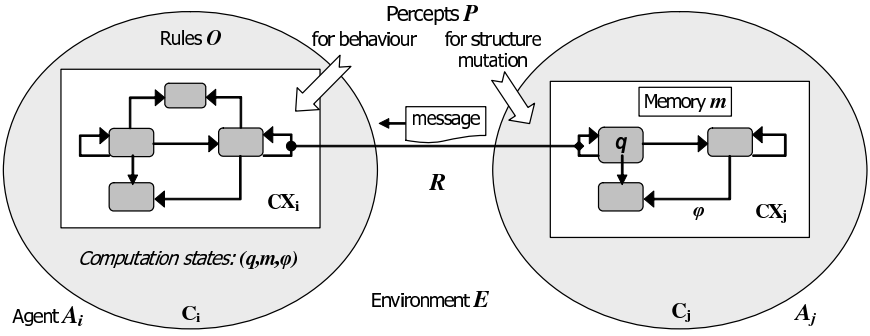


Fig. 3. An abstract example of a $OPERAS_{XC}$ consisting of two agents

communication rules also exist, as in PPS, so that there is indirect communication (through the environment) between the structural mutators (cells);

- $P = P_B \cup P_{SM}$ is the set of percepts of all participating agents, where $P_B = \Sigma_1 \cup \ldots \cup \Sigma_t$ is the set of inputs perceived by the XM model of the behaviour (subscript B) and $P_{SM} = (Q_1 \times M_1 \times \Phi_1) \cup \ldots \cup (Q_1 \times M_t \times \Phi_t)$ is the set of objects (alphabet) of the PPS mechanism that captures structure mutation (subscript SM), t being the number of types of agents;
- $E = \{(q, m, \varphi)_i | 1 \leq i \leq n, q \in Q_i, m \in M_i, \varphi \in \Phi_i\}$ holding information about the initial computation states of all the participating agents;
- $R : CXM \times CXM$ (CXM: the set of CXMs that model agent behaviour);
- $A = \{A_1, \ldots, A_n\}$ where $A_i = (CXM_k, C_k)_i$ is a particular agent of type k defined in terms of its individual behaviour (CXM_k) and its local structural mutator cell for controlling reconfiguration (C_k). The structural mutator cell is of the form $C_k = (w_i, o_k)$ where w_i is the multi-set of objects it contains and $o_k \subset O$ is the set of rules that correspond to the particular type of agent, k;
- $S = \{(XT_k, C_k) | \forall k \in Type\}$, where XT_k is an XM *type* (no initial state and memory).

The above mentioned operators attachment **ATT** and detachment **DET** have the same effect as the bond-making rules of a PPS, while the operators generation **GEN** and destruction **DES**, have the same effect as cell division an cell death of a PPS respectively. Formal definitions of these operators can be found in [33].

In this model, each structural mutator cell implicitly knows the computation state (q, m, φ) of the underlying XM that models behaviour. Environmental input is directed straight to the agent's behavioural part. In each computation cycle an input triggers a function of the behaviour CXM and the updated information about the agent's current computation state is updated in the structural mutator cell. A copy of the object is placed in the environment for other agents in the local environment to have access to it. Objects from the environment representing the computation states of neighbouring agents are imported and finally, all the reconfiguration rules in O of the type of the particular cell are being checked and if necessary applied. Since the model follows the computation rules

of a CXM system (triggered by the behaviour component's input, asynchronously for the different participating agents), computation of the behaviour-driven version of $OPERAS_{XC}$ is *asynchronous*. In another version of $OPERAS_{XC}$, the computation is cell-driven, and therefore *synchronous*. A detailed and more formal analysis of the two versions, however, falls outside the scope of this paper. In addition, as said previously, other instances of $OPERAS$ using these two methods, such as $OPERAS_{CC}$, $OPERAS_{XX}$ and $OPERAS_{CX}$ are possible but rather cumbersome.

4 OPERAS$_{XC}$ for a Swarm-Based System

4.1 Autonomous Spacecrafts for Asteroid Exploration

A representative example of a system which clearly possesses all the aforementioned characteristics of a dynamic MAS is the NASA Autonomous Nano-Technology Swarm (ANTS) system [23]. The NASA ANTS project aims at the development of a mission for the exploration of space asteroids with the use of different kinds of unmanned spacecrafts. Though each spacecraft can be considered as an autonomous agent, the successful exploration of an asteroid depends on the overall behaviour of the entire mission, as the latter emerges as a result of self-organisation. We chose this case study because correctness of the system has been identified as a primary requirement. Relevant work on the particular project included research on and comparison of a number of formal methods [23, 34], including CXMs.

The ANTS mission uses of three kinds of unmanned spacecrafts: leaders, L, (or rulers or coordinators), workers, W, and messengers, M (Fig. 4). The leaders are the spacecrafts that are aware of the goals of the mission and have a non-complete model of the environment. Their role is to coordinate the actions of the spacecrafts that are under their command but by no means should they be considered to be a central controlling mechanism as all spacecrafts' behaviour is autonomous. Depending on its goals, a leader creates a team consisting of a number of workers and at least one messengers. Workers and messengers are assigned to a leader upon request by (i) another leader, if they are not necessary for the fulfilment of its goals, or (ii) earth (if existing spacecrafts are not sufficient in number to cover current needs, new spacecrafts are allocated to the mission).

A worker is a spacecraft with a specialised instrument able, upon request from its leader, to take measurements from an asteroid while flying by it. It also possesses a mechanism for analysing the gathered data and sending the analysis results back to its leader in order for them to be evaluated. This in turn might update the view of the leader, i.e. its model of the environment, as well as its future goals.

The messengers, finally, are the spacecrafts that coordinate communication among workers, leaders and the control centre on earth. While each messenger is under the command of one leader, it may also assist in the communication of other leaders if its positioning allows it and conditions demand it.

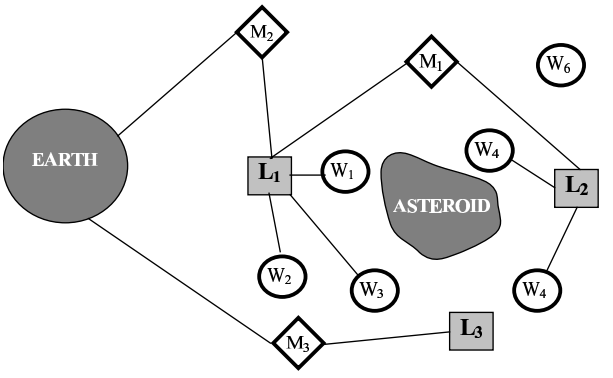


Fig. 4. An instance of the ANTS mission, L: Leader, W:Worker, M:Messenger

What applies to all types of spacecrafts is that in the case that there is a malfunctioning problem, their superiors are being notified. If the damage is irreparable they need to abort the mission while on the opposite case they may "heal" and return back to normal operation.

4.2 Leader: Formal Modelling of Behaviour in OPERAS$_{XC}$

The leader agent L can be modelled as an XM, whose state transition diagram F_L is depicted in Fig. 5. $Q_L = \{Processing, Malfunctioning, Aborting\}$ is the set of states a leader may be in. Its memory contains information about its current status (i.e. position and operational status), the IDs and statuses of the messengers and workers under its command, the analysis results up to this point, its current model of the surroundings as well as its goals: $M_L : \mathcal{S}tatus \times \mathbb{P}(\mathcal{M} \times \mathcal{S}tatus) \times \mathbb{P}(\mathcal{W} \times \mathcal{S}tatus) \times \mathcal{A}nalysisResults \times \mathcal{M}odel \times \mathcal{G}oals$ where $\mathcal{S}tatus : (Z \times Z \times Z) \times \{Q_L\}$ (Z being the set of positive integers, the 3-tuple denoting a position), $\mathbb{P}$ stands for power-set, $\mathcal{M}$ is the set of messengers, $\mathcal{W}$ is the set of workers and so forth.

The input set for the leader XM is $\Sigma_L = \{abrt, problem, remedy\} \cup (\mathcal{W} \times \mathcal{S}tatus) \cup (\mathcal{W} \times \mathcal{M}easurements) \cup (\{request, requestFromEarth, requestedFor\} \times \mathcal{I}nstrument)$, where $abrt$, $problem$, $remedy$, $request$, $requestedFor$ are constants and $\mathcal{I}nstrument$ is the set of containing the different types of installed instruments of the workers. The output set Γ_L is a set of informative messages.

Indicatively, some of the functions in the Φ_L set (functions are of the form: $function(input, memory\_tuple) = (output, memory\_tuple')$) are:

$acceptRequestForWorker\ ((requestedFor, instr), (\_, \_, workers, \_, \_, \_)) =$
$\quad('reassigned\ worker', (\_, \_, workers', \_, \_, \_))$
$\quad$ if $(w_i, (\_, \_, instr)) \in workers$
$\quad$ and $isWorkerNeeded(w_i) == false$
$\quad$ where $workers' = workers \backslash (w_i, (\_, \_, instr))$

$receiveWorker(w_i, (\_, \_, workers, \_, \_, \_)) =$
$\quad('received\ worker', (\_, \_, workers \cup (w_i), \_, \_, \_))$

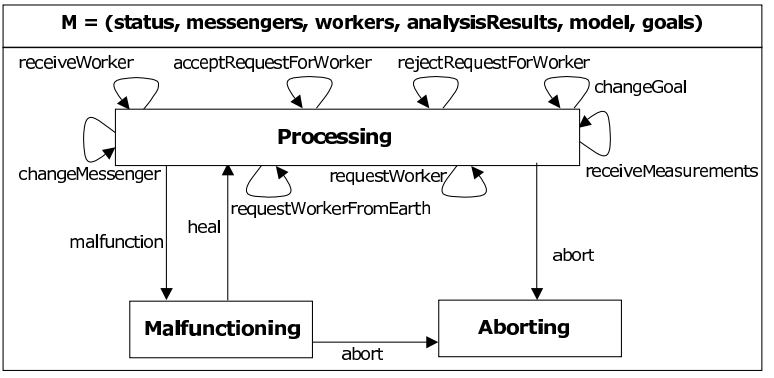


Fig. 5. State transition diagram of the Leader X-machine

As aforementioned, we used XMs for agent formal modelling because they facilitate formal verification and testing. These operations are crucial in developing mission critical systems. $\mathcal{X}m$CTL , an extension of CTL for XMs, can be used to verify models against the requirements, since it can prove that certain properties are true. Such properties are implicitly encoded in the memory structure of the XM model [30]. For example, the property "there exists a computation path in which a leader will accomplish all its goals and in all previous states the leader was employing at least one worker" is specified in $\mathcal{X}m$CTL as:

$$\mathbf{E}[\mathbf{M_x}(mem_L(3) \neq \emptyset)\ \mathbf{U}\ \mathbf{M_x}(mem_L(6) = \emptyset)]$$

where $mem_L(i)$ indicates the i-th element in the memory tuple of the leader model. Additionally, it is possible under certain well defined conditions, to produce a complete test set out of an XM model. The test set guarantees to determine the correctness of the implementation of each agent [31].

4.3 Worker: Formal Modelling of Behaviour in OPERAS$_{XC}$

The state transition diagram of the worker XM is depicted in Fig. 6. The internal states in which a worker may be are $Q_W = \{Measuring, Analysing, Malfunctioning, Aborting\}$ and its memory holds information about its current status (i.e. position, operational status and installed instrument), the identity and status of its commanding leader, the messengers which assist its communication, the target asteroid, the data received from the measurements and the results of the data analysis: $M_W : \mathcal{S}tatus \times (\mathcal{L} \times \mathcal{S}tatus) \times \mathbb{P}(\mathcal{M} \times \mathcal{S}tatus) \times \mathcal{T}arget \times \mathcal{M}easurements \times \mathcal{A}nalysisResults$.

The input set is $\Sigma_W = \{measure, analyse, send, abrt, problem, remedy\} \cup (\mathcal{L} \times \mathcal{S}tatus)$, where $abrt$, $problem$, $remedy$, $measure$, $analyse$ and $send$ are constants. The output set Γ_W is a set of informative messages.

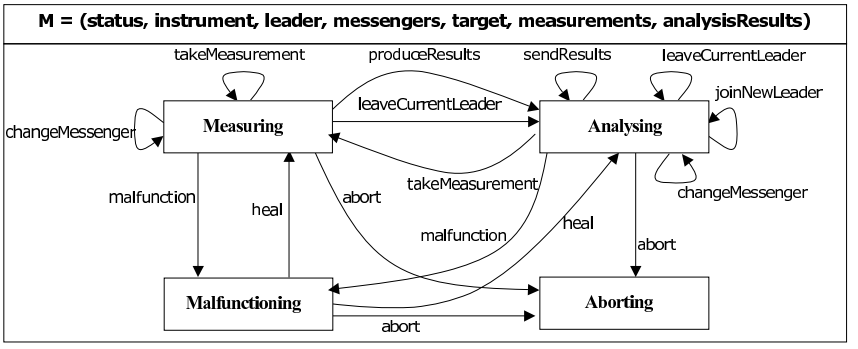


Fig. 6. State transition diagram of the Worker X-machine

Indicatively, some of the functions in the Φ_W set are:

$produceResults(analyse, (\_, \_, \_, \_, meas, analysisResults)) =$
$('analysed', (\_, \_, \_, \_, \emptyset, analysisResults'))$,
where $analysisResults' = analysisMechanism(meas) :: analysisResults$

$sendResults(send, (\_, \_, \_, \_, \_, res :: analysisResults)) =$
$('sent\ results', (\_, \_, \_, \_, \_, analysisResults))$

$leaveCurrentLeader((newLeader, st), (\_, (leader, st_0), \_, \_, \_, \_)) =$
$('been\ reassigned', (\_, newLeader, \_, \_, \_, \_))$

The model of the messenger agent is similarly created.

4.4 Formal Modelling of Structure Mutation in OPERAS$_{XC}$

According to $OPERAS_{XC}$, for the definition of the given system as a dynamic MAS, we need to assume an initial configuration. To keep the size restricted for demonstrative purposes, let us consider an initial configuration that includes one leader L_1, one messenger M_1 and two workers W_1, W_2.

The set O contains the following reconfiguration rules regarding: (a) generation of a new worker when the control centre on earth decides it should join the mission, (b) the destruction (i.e. removal from the system) of any kind of agent in the case it must abort the mission, (c) the establishment of a communication channel between a leader and a newly assigned to it worker, and (d) the removal of a communication channel between a leader and a worker when the latter is being reassigned to a new leader.

More particularly O contains the following rules:

If there is a need for an additional worker and earth can allocate one than a new agent appear in system ANTS:

$(\_, \_, requestWorkerFromEarth)_{L_i} \wedge earthHasAvailableWorkers() == true$
$\Rightarrow \mathbf{GEN}(W_i, q_{0_i}, m_{0_i}, ANTS)_L$

If an agent aborts the mission then the agent is removed from system ANTS:

$(aborting, \_, \_)_{*_{this}} \Rightarrow \mathbf{DES}(*_{this}, ANTS)_*$

If a worker agent looses its contact with its leader then the communication channels between the two agents are broken:

$(\_, (\_, \_, L_i, \_, \_, \_, \_), leaveCurrentLeader)_{W_i}$
$\Rightarrow \mathbf{DET}(W_i, L_i, \mathbf{DET}(L_i, W_i, ANTS))_W$

If a worker agent is assigned with a new leader then a new communication channel is established:

$(\_, (\_, \_, newLeader, \_, \_, \_, \_), joinNewLeader)_{W_i}$
$\Rightarrow \mathbf{ATT}(W_i, newLeader, ANTS)_W$

If a leader agent is assigned with a new worker (either from another leader or from earth) then a new communication channel is established:

$(\_, (\_, \_, newWorker :: workers, \_, \_, \_), receiveWorker)_{L_i}$
$\Rightarrow \mathbf{ATT}(L_i, newWorker, ANTS)_L$

$(\_, (\_, \_, newWorker :: workers, \_, \_, \_), receiveWorkerFromEarth)_{L_i}$
$\Rightarrow \mathbf{ATT}(L_i, newWorker, ANTS)_L$

where * stands for any type of agent.

The set of percepts of all agents is:

$$P = \Sigma_L \cup \Sigma_W \cup \Sigma_M \cup (Q_L \times M_L \times \Phi_L) \cup (Q_W \times M_W \times \Phi_W) \cup (Q_M \times M_M \times \Phi_M).$$

Because all reconfiguration rules per type of agent rely only on conditions dependent on the computation state of the agent itself (and not other agents), the model needs not to communicate computation states among the different agents and there are, therefore, no additional communication rules. A direct consequence of this is that there is no need for the environment to play the role of communication mediator between the participating entities and as such no need for it to hold any computation state objects: $E = \emptyset$.

Since in the assumed initial configuration we consider to have one group of spacecrafts under the command of one leader, all agents should be in communication with all others and so:

$R = \{(L_1, W_1),\ (L_1, W_2),\ (L_1, M_1),\ (M_1, L_1),\ (M_1, W_1),\ (M_1, W_2),\ (W_1, L_1),\ (W_1, M_1),\ (W_2, L_1),\ (W_2, M_1)\}$

The set that contains all the agent instances becomes: $A = \{L_1, W_1, W_2, M_1)\}$ where $L_1 = (CXM_{L_1}, C_{L_1})$, $W_i = (CXM_{W_i}, C_{W_i}), 1 \leq i \leq 2$ and $M_1 = (CXM_{M_1}, C_{M_1})$.

Finally, the set S that contains the "genetic codes" for all agent types is:

$S = \{(XT_L, C_L), (XT_W, C_W), (XT_M, C_M)\}$ where L, W, M are the XMs defined previously.

5 Conclusions and Further Work

We presented *OPERAS*, a framework, with which one can formally model the behaviour and control over the internal states of an agent as well as formally describe the mutations that occur in the structure of a MAS, as two separate

components. Driven by a formal methods perspective, we employed CXMs and ideas from PPSs to define $OPERAS_{XC}$, a particular instance of the framework. These gave us the opportunity to combine the advantages that XMs have in terms of modelling the behaviour of an agent, testing it and verifying its properties with the advantages that PPSs have in terms of defining the mutation of the structure of a MAS. We have experimented with modelling of various biological and biology-inspired systems. In this paper we presented the $OPERAS_{XC}$ model of a swarm-based system of a number of autonomous spacecrafts, a case which has been used by researchers for comparative study of formal methods.

We would like to continue the investigation of how $OPERAS$ could employ other formal methods that might be suitable for this purpose. In the near future, we will focus on theoretical aspects of the framework, in order to demonstrate its usefulness towards developing correct agent societies (i.e. complete testing and verification). Although work on verification and testing has been done with XMs [30, 31], it is important to investigate to what extent this could be inherited in a hybrid system, like $OPERAS_{XC}$. Towards this direction, we are also currently working on various types of transformations that could prove its power for formal modelling as well as address legacy issues with respect to correctness [35]. These developments are mainly of interest to the formal method community.

On the other hand, the MAS community might be interested in how $OPERAS_{XC}$ can facilitate the implementation of agent systems. Towards this end, we started our efforts to achieve integration of existing development tools on XMs and PPSs in order to come up with a new tool that will be able to initially animate $OPERAS_{XC}$ specified models. The integration of the necessary features of these two tools into one will allow us to gain a deeper understanding of the modelling issues involved in engineering agent societies with $OPERAS_{XC}$ and help us investigate the practicability of our approach.

Acknowledgements

The authors would like to thank the reviewers for their valuable initial and additional comments as well as the Hellenic Artificial Intelligence Society for funding our participation to the ESAW 2007 workshop.

References

[1] Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case studies for self-organization in computer science. Journal of Systems Arch. 52, 443–460 (2006)
[2] Dignum, V., Dignum, F.: Understanding organizational congruence: Formal model and simulation framework. In: Proceedings of the Agent-Directed Simulation Symposium (ADS 2007), Norfolk, USA (March 2007)
[3] Dignum, V., Dignum, F.: A logic for agent organization. In: Proceedings of the Workshop on Formal Approaches to Multi-Agent Systems Durham, September 3-7 (2007)

[4] Hoogendoorn, M., Schut, M.C., Treur, J.: Modeling decentralized organizational change in honeybee societies. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 615–624. Springer, Heidelberg (2007)
[5] Charrier, R., Bourjot, C., Charpillet, F.: Deterministic nonlinear modeling of ant algorithm with logistic multiagent system. In: Proceedings of the 6th international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007). ACM, New York (2007)
[6] Matson, E., DeLoach, S.: Formal transition in agent organizations. In: Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp. 235–240 (2005)
[7] DeLoach, S.A.: Engineering organization-based multiagent systems. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 109–125. Springer, Heidelberg (2006)
[8] dInverno, M., Luck, M., Georgeff, M., Kinny, D., Wooldridge, M.: The dMARS architechure: A specification of the distributed multi-agent reasoning system. Autonomous Agents and Multi-Agent Systems 9, 5–53 (2004)
[9] Rabinovich, Z., Rosenschein, J.S.: Dynamics based control: Structure. In: Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains, at The 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, pp. 148–161 (2006)
[10] Luck, M., d'Inverno, M.: Formal methods and agent-based systems. In: Rouff, C., Truszkowski, M.H.J.R.J., Gordon-Spears, D. (eds.) NASA Monographs in Systems and Software Engineering. Springer, Heidelberg (2006)
[11] Fisher, M., Wooldridge, M.: On the formal specification and verification of multi-agent systems. International Journal of Cooperating Information Systems 6, 37–65 (1997)
[12] Hilaire, V., Koukam, A., Gruer, P., Müller, J.P.: Formal specification and prototyping of multi-agent systems. In: Omicini, A., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2000. LNCS (LNAI), vol. 1972, pp. 114–127. Springer, Heidelberg (2000)
[13] Odell, J., Parunak, H.V.D., Bauer, B.: Extending UML for agents. In: Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, pp. 3–17 (2000)
[14] Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multiagent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
[15] Gutknecht, O., Ferber, J.: MadKit: a generic multi-agent platform. In: Proc. of the 4th International Conference on Autonomous Agents, pp. 78–79 (2000)
[16] Chopra, A.K., Mallya, A.U., Desai, N.V., Singh, M.P.: Modeling flexible business processes. In: AAMAS 2004 (2004)
[17] Krishna, S.N., Păun, G.: P systems with mobile membranes. Natural Computing: an international journal 4, 255–274 (2005)
[18] Cardelli, L., Gordon, A.D.: Mobile ambients. In: Nivat, M. (ed.) FOSSACS 1998. LNCS, vol. 1378, pp. 140–155. Springer, Heidelberg (1998)
[19] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes I. Information and Computation 100, 1–40 (1992)
[20] Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61, 108–143 (2000); Also circulated as a TUCS report since (1998)
[21] Banatre, J., Le Metayer, D.: The gamma model and its discipline of programming. Science of Computer Programming 15, 55–77 (1990)

[22] Berry, G., Boudol, G.: The chemical abstract machine. Journal of Theoretical Computer Science 96, 217–248 (1992)
[23] Rouf, C., Vanderbilt, A., Truszkowski, W., Rash, J., Hinchey, M.: Verification of NASA emergent systems. In: Proceedings of the 9th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2004), pp. 231–238 (2004)
[24] Stamatopoulou, I., Kefalas, P., Gheorghe, M.: Modelling the dynamic structure of biological state-based systems. BioSystems 87, 142–149 (2007)
[25] Kefalas, P., Stamatopoulou, I., Gheorghe, M.: A formal modelling framework for developing multi-agent systems with dynamic structure and behaviour. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 122–131. Springer, Heidelberg (2005)
[26] Stamatopoulou, I., Kefalas, P., Gheorghe, M.: Specification of reconfigurable MAS: A hybrid formal approach. In: Antoniou, G., Potamias, G., Spyropoulos, C., Plexousakis, D. (eds.) SETN 2006. LNCS (LNAI), vol. 3955, pp. 592–595. Springer, Heidelberg (2006)
[27] Stamatopoulou, I., Kefalas, P., Gheorghe, M.: OPERAS$_{CC}$: An instance of a formal framework for MAS modelling based on Population P Systems. In: Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2007. LNCS, vol. 4860, pp. 551–566. Springer, Heidelberg (2007)
[28] Eilenberg, S.: Automata, Languages and Machines. Academic Press, London (1974)
[29] Kefalas, P., Eleftherakis, G., Kehris, E.: Communicating X-machines: A practical approach for formal and modular specification of large systems. Journal of Information and Software Technology 45, 269–280 (2003)
[30] Eleftherakis, G.: Formal Verification of X-machine Models: Towards Formal Development of Computer-based Systems. PhD thesis, Department of Computer Science, University of Sheffield (2003)
[31] Holcombe, M., Ipate, F.: Correct Systems: Building a Business Process Solution. Springer, London (1998)
[32] Bernandini, F., Gheorghe, M.: Population P Systems. Journal of Universal Computer Science 10, 509–539 (2004)
[33] Kefalas, P., Eleftherakis, G., Holcombe, M., Stamatopoulou, I.: Formal modelling of the dynamic behaviour of biology-inspired agent-based systems. In: Gheorghe, M. (ed.) Molecular Computational Models: Unconventional Approaches, pp. 243–276. Idea Publishing Inc. (2005)
[34] Rouff, C., Vanderbilt, A., Hinchey, M., Truszkowski, W., Rash, J.: Properties of a formal method for prediction of emergent behaviors in swarm-based systems. In: Procedings of the 2nd International Conference on Software Engineering and Formal Methods, pp. 24–33 (2004)
[35] Kefalas, P., Stamatopoulou, I., Gheorghe, M.: Principles of transforming Communicating X-machines to Population P Systems. In: Proceedings of the International Workshop on Automata for Cellular and Molecular Computing (ACMC 2007) (2007); Also to appear in the International Journal of Foundations of Computer Science

The Acquisition of Linguistic Competence for Communicating Propositional Logic Sentences*

Josefina Sierra[1] and Josefina Santibáñez[2]

[1] Departamento de Lenguajes y Sistemas Informáticos
Universidad Politécnica de Cataluña, Spain
jsierra@lsi.upc.edu
[2] Departamento de Ciencias de la Educación
Universidad de La Rioja, Spain
josefina.santibanez@unirioja.es

Abstract. We describe some experiments which show how a *language* expressive enough to allow the communication of meanings of the same complexity as propositional logic formulas can emerge in a population of autonomous agents which have no prior linguistic knowledge. We take an approach based on *general purpose cognitive capacities,* such as invention, adoption and induction, and on *self-organisation* principles applied to a particular type of linguistic interaction known as a *language game*.

These experiments extend previous work by considering a larger population and a much larger search space of grammar rules. In particular the agents are allowed to order the expressions associated with the constituents of a logical formula in arbitrary order in the sentence. Previous work assumed that the expressions associated with the connectives should be always placed in the first position of the sentence. Another difference is that communication is considered successful in a language game if the meaning interpreted by the hearer is *logically equivalent* to the meaning the speaker had in mind. In previous experiments the meanings of speaker and hearer were required to be syntactically equal. This allows us to observe how a less strict grammar in terms of word order emerges through the self-organisation process, which minimizes the learning effort of the agents by imposing only those order relations among the components of a sentence that are necessary for language understanding.

1 Introduction

This paper addresses the problem of the acquisition of a language (i.e., a lexicon and a grammar) expressive enough to allow the communication of meanings that can be represented by propositional logic formulas. We take an approach based on *general purpose cognitive capacities,* such as invention, adoption and induction. Coordination of the linguistic knowledge acquired by the individual agents is achieved through a *self-organisation* process of the linguistic interactions that take place between pairs of agents of the population.

* This work is partially funded by the DGICYT TIN2005-08832-C03-03 project (MOISES-BAR).

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 175–192, 2008.

We describe some experiments which show how a shared set of preferred lexical entries, syntactic categories and grammatical constructions (i.e., a *language*) can emerge in a population of autonomous agents which have no prior linguistic knowledge. This shared language is expressive enough to allow the agents to communicate any meaning that can be represented by a propositional logic formula.

These experiments extend previous work [1] by considering a larger population and a much larger search space of grammar rules. In particular the agents are allowed to order the expressions associated with the constituents of a logical formula in arbitrary order in the sentence. Previous work assumed that the expressions associated with the connectives should be always placed in the first position of the sentence. The branching factor of the search space of grammar rules that can be used for expressing formulas constructed with binary connectives is extended thus from two to six.

Another difference is that communication is considered successful in a language game if the meaning interpreted by the hearer is *logically equivalent* to the meaning the speaker had in mind. In previous experiments both meanings were required to be syntactically equal, i.e., the same formula. This allows us to observe how a less strict grammar in terms of word order emerges through the self-organisation process, which minimizes the learning effort of the agents by imposing only those order relations among the components of a sentence that are necessary for language understanding.

To understand how a population of autonomous agents might be able to come up with a language expressive enough to communicate propositional logic formulas is a problem of practical and theoretical interest. The important role of logic as a formalism for *knowledge representation* and *reasoning* [2] is well known in *artificial intelligence*. Much of the knowledge used by artificial intelligent agents today is represented in logic. In particular the recent development of efficient algorithms for checking satisfiability (*SAT solvers*) is increasing the number of practical applications that use propositional logic as its knowledge representation formalism. Logic is relevant as well for computational and cognitive *linguistics,* because it is the standard formalism used in these fields for representing semantic information (i.e., the meanings of words and sentences). On the other hand logical connectives and logical constructions are themselves a fundamental part of *natural language*, and they play a very important role in the *development of intelligence* and deductive reasoning [3,4,5,6]. Therefore from a scientific point of view it is necessary to understand how an agent can both conceptualise and communicate logical constructions to other agents.

The research presented in this paper assumes previous work on the conceptualisation of logical connectives [7,8]. In [9] a grounded approach to the acquisition of logical categories (i.e., logical connectives) based on the discrimination of a "subset of objects" from the rest of the objects in a given context is described. The "subset of objects" is characterized by a logical formula constructed from perceptually grounded categories. This formula is satisfied by the objects in the subset and not satisfied by the rest of the objects in the context. In this paper we only focus on the problem of the acquisition of a language (a vocabulary and

a grammar) suitable for expressing propositional logic formulas. Future work will address the complete problem with which children are faced which consists in acquiring both the semantics and the syntax of the logical constructions and connectives that are used in natural language.

The rest of the paper is organised as follows. First we introduce the formalism used for representing the grammars constructed by the agents. Then we describe the particular type of language game played by the agents, focusing on the main cognitive processes they use for constructing a shared lexicon and a grammar: invention, adoption, induction and co-adaptation. Next we present the results of some experiments in which a population of autonomous agents constructs a language that allows communicating propositional logic formulas. Finally we summarize some related work and the contributions of the paper.

2 Grammatical Formalism

The formalism used for representing the grammars constructed by the agents is *Definite Clause Grammar*. In particular *non-terminals* have three arguments with the following contents: (1) semantic information; (2) a *score* in the interval [0, 1] that estimates the usefulness of the rule in previous communication; and (3) a *counter* that records the number of times the rule has been used.

Let us consider some examples of grammars the agents could use to express the propositional formula $right \wedge light$[1]. The first grammar consists of a single rule which states that 'andrightlight' is a valid sentence meaning $right \wedge light$.

$$s([and, right, light]), S) \rightarrow andrightlight, \{S\ is\ 0.01\} \tag{1}$$

The same formula could be expressed as well using the following compositional, recursive grammar: s is the start symbol, $c2$ is the name of a syntactic category associated with binary connectives. Like in Prolog, variables start with a capital letter and constants with a lower case letter.

The number that appears in first place on the right hand side of a grammar rule (see rule 5) indicates the position of the expression associated with the connective in the sentence: The number 1 indicates that the expression associated with the connective is a *prefix* (first position), number 2 that it is an *infix* (second position), and number 3 that it is a *suffix* (third position). We use this convention because Prolog does not allow the use of left recursive grammar rules.

$$s(\text{light}, S) \rightarrow \text{light}, \{S\ is\ 0.70\} \tag{2}$$

$$s(\text{right}, S) \rightarrow \text{right}, \{S\ is\ 0.25\} \tag{3}$$

$$c2(\text{and}, S) \rightarrow \text{and}, \{S\ is\ 0.50\} \tag{4}$$

[1] Notice that we use Prolog grammar rules for describing the grammars. The semantic argument of non-terminals uses Lisp like (prefix) notation for representing propositional formulas (e.g., the Prolog list $[and, [not, right], light]$ is equivalent to $\neg right \wedge light$). The third argument (the use counter) of non-terminals is not shown in the examples.

$$s([P,Q,R],S) \rightarrow 2,\ c2(P,S1), s(Q,S2), s(R,S3), \{S\ is\ S1 \cdot S2 \cdot S3 \cdot 0.01\} \quad (5)$$

This grammar breaks down the sentence 'rightandlight' into subparts with independent meanings. The whole sentence is constructed concatenating these subparts. The meaning of the sentence is composed combining the meanings of the subparts using the variables P, Q and R.

The agents can invent a large number of grammars to express the same formula, because they can associate different words with the propositional constants and connectives of the formula, and they can concatenate the expressions associated with the constituents of the formula in any order. The following grammar uses the sentence 'claroderechay' for expressing the same formula $right \wedge light$.

$$s(\text{light}, S) \rightarrow \text{claro}, \{S\ is\ 0.60\} \quad (6)$$

$$s(\text{right}, S) \rightarrow \text{derecha}, \{S\ is\ 0.40\} \quad (7)$$

$$c2(\text{and}, S) \rightarrow \text{y}, \{S\ is\ 0.50\} \quad (8)$$

$$s([P,Q,R],S) \rightarrow 3,\ c2(P,S1), s(R,S2), s(Q,S3), \{S\ is\ S1 \cdot S2 \cdot S3 \cdot 0.01\} \quad (9)$$

Coordination of the grammars constructed by the individual agents is therefore not a trivial task, because in order to understand each other the agents must use a common vocabulary and must order the constituents of compound sentences in sufficiently similar ways as to avoid ambiguous interpretations.

3 Language Games

Language acquisition is seen thus as a collective process by which a population of autonomous agents constructs a *common language* that allows them to communicate some set of meanings. Such an agreement on the agents' vocabularies and individual grammars is achieved through a process of self-organisation of the linguistic interactions that take place among the agents in the population.

In the experiments described in this paper the agents interact with each other playing language games. A *language game* [10,11], which is played by a pair of agents randomly chosen from the population, consists of the following actions:

1. The speaker chooses a formula (i.e., a meaning) from a given propositional language, generates or invents a sentence that expresses this formula, and communicates that sentence to the hearer.
2. The hearer tries to interpret the sentence communicated by the speaker. If it can parse it using its lexicon and grammar, it extracts a meaning (i.e., a formula) which can be logically equivalent or not to the formula intended by the speaker. If the hearer cannot parse the sentence, the speaker communicates

the formula it had in mind to the hearer, and the hearer adopts an association between the formula and the sentence used by the speaker[2].
3. Depending on the outcome of the language game both agents adjust their grammars in order to become more successful in future language games.

3.1 Invention

The agents in the population start with an empty lexicon and grammar. Therefore they cannot generate sentences for most meanings at the early stages of a simulation run. In order to allow language to get off the ground, they are allowed to invent new sentences for those meanings they cannot express using their lexicons and grammars in the first step of a language game.

The invention algorithm generates a sentence E for a propositional formula F as follows. If F is atomic, it invents a new word E[3]. If F is a formula constructed using a connective (it is of the form $\neg A$ or $A \otimes B$), it generates an expression for the connective and for each subformula of F using the agent's grammar if it can, or inventing a new one if it cannot, and it concatenates these expressions randomly in order to construct a sentence E for the whole meaning F.

As the agents play language games they learn associations between expressions and meanings, and induce linguistic knowledge from such associations in the form of grammatical rules and lexical entries. Once they can generate sentences for expressing a particular meaning using their own grammars, they select the sentence with the highest score and communicate that sentence to the hearer. The algorithm for computing the score of a sentence from the scores of the grammatical rules used in its generation is explained in detail later.

3.2 Adoption

In the second step of a language game the hearer tries to interpret the sentence communicated by the speaker. If it can parse it using its lexicon and grammar it extracts a meaning, and checks whether its interpretation is right or wrong (i.e., it is logically equivalent to the meaning intended by the speaker) in the third step of the language game. However at the early stages of a simulation run the agents usually cannot parse the sentences communicated by the speakers, since they have no prior linguistic knowledge. In this case the speaker communicates the formula F it had in mind to the hearer, and the hearer adopts an association between that formula and the sentence E used by the speaker adding a new rule of the form $s(F, S) \rightarrow E, \{S\ is\ 0.01\}$ to its grammar[4].

At later stages of a simulation run it usually happens that the grammars and lexicons of speaker and hearer are not consistent, because each agent constructs

[2] A language game *succeeds* if the hearer can parse the sentence communicated by the speaker and it extracts a meaning (i.e., a formula) that is logically equivalent to the formula the speaker had in mind; otherwise the language game *fails*.
[3] New words are sequences of one to three letters randomly chosen from the alphabet.
[4] The score of the rules generated using invention, adoption or induction is initialized to 0.01.

its own grammar from the linguistic interactions in which it participates, and it is very unlikely that speaker and hearer share the same history of linguistic interactions unless the population consists only of these two agents. In this case the hearer may be able to parse the sentence generated by the speaker, but its interpretation of that sentence might be different from the meaning the speaker had in mind. The strategy used to coordinate the grammars of speaker and hearer when this happens is to decrease the score of the rules used by speaker and hearer in the processes of generation and parsing, respectively, and allow the hearer to adopt an association between the sentence and the meaning used by the speaker. Adoption however does not always take place in this case, because it is possible that the hearer knows the grammatical rules used by the speaker, but the scores of these rules are not higher than the scores of the rules it used for interpretation. The hearer adopts only an association between a sentence and a meaning if it cannot generate such an association using its lexicon and grammar.

3.3 Induction

Besides inventing and adopting associations between sentences and meanings, the agents can use some *induction mechanisms* to extract generalizations from the grammar rules they have learnt so far [12,13]. The induction mechanisms used in this paper are based on the rules for simplification and chunk in [14,15], although we have extended them so that they can be applied to grammar rules which have scores and which mark with a number the position of the connective in the sentence. We use the approach proposed in [16] for computing the scores of sentences and meanings from the scores of the rules used in their generation.

The induction rules are applied whenever the agents invent or adopt a new association to avoid redundancy and increase generality in their grammars.

Simplification. *Let* $r1$ *and* $r2$ *be a pair of grammar rules such that the semantic argument of the left hand side of* $r1$ *contains a subterm* $m1$*,* $r2$ *is of the form* $n(m1, S) \rightarrow e1, \{S \; is \; C1\}$*, and* $e1$ *is a substring of the terminals of* $r1$*. Then simplification can be applied to* $r1$ *replacing it with a new rule that is identical to* $r1$ *except that: (1)* $m1$ *is replaced with a new variable* X *in the semantic argument of the left hand side; (2)* $e1$ *is replaced with* $n(X, S)$ *on the right hand side; and (3) the arithmetic expression* $\{R \; is \; E \cdot C2\}$ *on the right hand side of* $r1$ *is replaced with a new arithmetic expression of the form* $\{R \; is \; E \cdot S \cdot 0.01\}$*, where* $C1$ *and* $C2$ *are constants in the range [0,1], and* E *is the product of the score variables that appeared on the right hand side of* $r1$*.*

Let us see how simplification works with an example. Suppose an agent's grammar contains rules 2 and 3. It plays a language game with another agent, and invents or adopts the following rule.

$$s([and, light, right], S) \rightarrow lightandright, \{S \; is \; 0.01\}. \tag{10}$$

It could apply simplification to rule 10 (using rule 3) and replace it with 11.

$$s([and, light, R], S) \rightarrow lightand, s(R, SR), \{S \; is \; SR \cdot 0.01\} \tag{11}$$

Now rule 11 could be simplified again, replacing it with 12 which contains specific information about the position of the connective in the sentence.

$$s([and, Q, R], S) \rightarrow 2, and, s(Q, SQ), s(R, SR), \{S\ is\ SQ \cdot SR \cdot 0.01\} \quad (12)$$

If later on the agent invents or adopts a rule that associates the sentence 'lightorright' with the formula $[or, light, right]$ and applies simplification, then its grammar would contain the following rules that are compositional and recursive, but which do not use a syntactic category for binary connectives.

$$s([and, Q, R], S) \rightarrow 2, and, s(Q, SQ), s(R, SR), \{S\ is\ SQ \cdot SR \cdot 0.01\} \quad (13)$$
$$s([or, Q, R], S) \rightarrow 2, or, s(Q, SQ), s(R, SR), \{S\ is\ SQ \cdot SR \cdot 0.01\} \quad (14)$$

Chunk I. *Let* $r1$ *and* $r2$ *be a pair of rules with the same left hand side category symbol. If the semantic arguments of the left hand sides of the rules differ only in one subterm* $m1$ *and* $m2$*, and there exist two strings of terminals* $e1$ *and* $e2$ *that, if replaced with the same non-terminal, would make the right hand sides of the rules identical, chunk can be applied as follows. A new category symbol* c *is created and the following new rules are added to the grammar.*

$$c(m1, S) \rightarrow e1, \{S\ is\ 0.01\} \qquad c(m2, S) \rightarrow e2, \{S\ is\ 0.01\}$$

Rules $r1$ *and* $r2$ *are replaced by a single rule that is identical to* $r1$ *except that: (1)* $m1$ *is replaced with a new variable* X *in the semantic argument of the left hand side; (2)* $e1$ *is replaced with* $c(X, S)$ *on the right hand side; and (3) the arithmetic expression* $\{R\ is\ E \cdot C1\}$ *on the right hand side of* $r1$ *is replaced with a new arithmetic expression of the form* $\{R\ is\ E \cdot S \cdot 0.01\}$*, where* $C1$ *is a constant in the range [0,1] and* E *is the product of the score variables that appeared on the right hand side of* $r1$*.*

The agent of previous examples could apply chunk I to rules 13 and 14 generating a new syntactic category $c2$ for binary connectives as follows.

$$s([P, Q, R], S) \rightarrow 2, c2(P, S1), s(Q, S2), s(R, S3), \{S\ is\ \mathrm{S1} \cdot \mathrm{S2} \cdot \mathrm{S3} \cdot 0.01\} \quad (15)$$
$$c2(\mathrm{and}, S) \rightarrow \mathrm{and}, \{S\ is\ 0.01\} \quad (16)$$
$$c2(\mathrm{or}, S) \rightarrow \mathrm{or}, \{S\ is\ 0.01\} \quad (17)$$

Rules 13 and 14 would be replaced with rule 15, which generalises them because it can be applied to formulas constructed using any binary connective, and rules 16 and 17, which state that the expressions *and* and *or* belong to $c2$ (the syntactic category of binary connectives[5]), would be added to the grammar.

[5] The syntactic category $c2$ is in fact more specific, as we shall see in section 4. It corresponds to binary connectives whose expressions are placed in the second position of the sentence, preceded by the expression associated with their first argument, and followed by the expression associated with their second argument.

Chunk II. *If the semantic arguments of the left hand sides of two rules* $r1$ *and* $r2$ *can be unified applying substitution* $X/m1$ *to* $r1$*, and there exists a string of terminals* $e1$ *in* $r2$ *that corresponds to a nonterminal* $c(X, S)$ *in* $r1$*, then rule* $r2$ *can be replaced by a new rule of the form* $c(m1, S) \rightarrow e1, \{S\ is\ 0.01\}$.

Suppose the agent of previous examples adopts or invents the following rule.

$$s([\text{iff}, light, right], S) \rightarrow \text{lightiffright}, \{S\ is\ 0.01\} \tag{18}$$

Simplification of rule 18 with rules 2 and 3 would replace rule 18 with 19.

$$s([\text{iff}, Q, R], S) \rightarrow 2, \text{iff}, s(Q, SQ), s(R, SR), \{S\ is\ SQ \cdot SR \cdot 0.01\} \tag{19}$$

Then chunk II, applied to 19 and 15, would replace rule 19 with rule 20.

$$c2(\text{iff}, S) \rightarrow \text{iff}, \{S\ is\ 0.01\} \tag{20}$$

3.4 Co-adaptation

Coordination of the grammars constructed by the individual agents is not a trivial task, because in order to understand each other the agents must use a common vocabulary and must order the constituents of compound sentences in sufficiently similar ways as to avoid ambiguous interpretations. Such an agreement on the agents' vocabularies and on their individual grammars is achieved through a process of self-organisation of the linguistic interactions that take place among the agents in the population.

It is necessary to coordinate the agents' grammars because different agents can invent different expressions for referring to the same propositional constants and connectives, and because the invention process uses a random order to concatenate the expressions associated with the components of a given meaning. Let us consider an example that illustrates the problem. Imagine that an agent has invented or adopted the following rules for expressing the meaning [if,light,right].

$$s([\text{if}, \text{light}, \text{right}], S) \rightarrow \textit{lightrightif}, \{S\ is\ 0.01\}$$
$$s([\text{if}, \text{light}, \text{right}], S) \rightarrow \textit{rightlightif}, \{S\ is\ 0.01\}$$

Simplification with rules 2 and 3 would replace them with the following rules which not only cannot be used for generating a syntactic category for implications (because they do not satisfy the preconditions of chunk I), but that are in fact *incompatible* because they associate the same sentence with two meanings which are not logically equivalent (they reverse the direction of the implication).

$$S([\text{if}, X, Y], SC) \rightarrow 3, \text{if}, s(X, SX), s(Y, SY), \{SC\ is\ SX \cdot SY \cdot 0.01\}$$

$$S([\text{if}, X, Y], SC) \rightarrow 3, \text{if}, s(Y, SY), s(X, SX), \{SC\ is\ SY \cdot SX \cdot 0.01\}$$

The agent would be forced thus to make a choice between one of these two rules in order to express implications in a consistent manner, and would try to choose the rule that is understood by most agents in the population.

Self-organisation mechanisms help to coordinate the agents' grammars in such a way that they prefer to use the rules that are used more often by other agents [17,18,19,20]. Coordination in the experiments takes place at the third stage of a language game, when the speaker communicates the meaning it had in mind to the hearer. Depending on the outcome of a language game speaker and hearer take different actions. We have explained some of them already (invention and adoption), but they *co-adapt their grammars* as well adjusting the scores of their rules in order to become more successful in future language games.

We consider first the case in which the speaker can generate a sentence for the meaning using the rules in its grammar. If the speaker can generate several sentences for expressing that meaning, it chooses the sentence with the highest score. The rest of the sentences are called *competing sentences*.

The *score of a sentence* (or a *meaning*) generated using a grammar rule is computed using the arithmetic expression on the right hand side of that rule. Consider the generation of a sentence for expressing the meaning $[and, right, light]$ using the following rules.

$$s(\text{light}, S) \rightarrow \text{light}, \{S\ is\ 0.70\} \tag{21}$$

$$s(\text{right}, S) \rightarrow \text{right}, \{S\ is\ 0.25\} \tag{22}$$

$$c2(\text{and}, S) \rightarrow \text{and}, \{S\ is\ 0.50\} \tag{23}$$

$$s([P, Q, R], S) \rightarrow 1, c2(P, S1), s(Q, S2), s(R, S3), \{S\ is\ S1 \cdot S2 \cdot S3 \cdot 0.01\} \tag{24}$$

The score S of the sentence *andrightligth*, generated by rule 24, is computed multiplying the score of that rule (0.01) by the scores of the rules 23, 22 and 21 which generate the substrings of that sentence (0.50, 0.25 and 0.70, respectively). The *score of a grammar rule* is the last number of the arithmetic expression that appears on the right hand side of that rule.

Suppose the hearer can interpret the sentence communicated by the speaker. If the hearer can obtain several meanings for that sentence, the meaning with the highest score is selected. The rest of the meanings are called *competing meanings*.

If the meaning interpreted by the hearer is logically equivalent to the meaning the speaker had in mind, the game succeeds and both agents adjust the scores of the rules in their grammars. The speaker increases the scores of the rules it used for generating the sentence communicated to the hearer and decreases the scores of the rules it used for generating competing sentences. The hearer increases the scores of the rules it used for obtaining the meaning the speaker had in mind and decreases the scores of the rules it used for obtaining competing meanings. This way the rules that have been used successfully get reinforced, and the rules that have been used for generating competing sentences or meanings are inhibited.

If the meaning interpreted by the hearer is not logically equivalent to the meaning the speaker had in mind, the game fails, and both agents decrease the scores

of the rules they used for generating and interpreting the sentence, respectively. This way the rules that have been used without success are inhibited.

The scores of grammar rules are *updated* using the scheme proposed in [10]. The rule's original score S is replaced with the result of evaluating expression 25 if the score is *increased*, and with the result of evaluating expression 26 if the score is *decreased*. The constant μ is a learning parameter which is set to 0.1.

$$minimum(1,\ S + \mu) \tag{25}$$

$$maximum(0,\ S - \mu) \tag{26}$$

A mechanism for **forgetting rules** that have not been useful in past language games is introduced to simplify the agents' grammars and avoid sources of ambiguity. Every ten language games the rules which have been used more than thirty times and have scores lower than 0.01 are removed from the grammars.

4 Experiments

We describe the results of some experiments in which a population of five agents constructs a common vocabulary and a grammar that allows communicating a set of meanings which corresponds to all the formulas of a propositional logic language.

In the experiments we have taken an incremental learning approach in which the agents first play 10010 language games about propositional constants, and then they play 15010 language games about logical formulas constructed using unary or binary connectives. At the end of a typical simulation run all the agents prefer the same expressions for naming the propositional constants of the language. The individual grammars built by the agents at the end of a typical simulation run (see table 1), although different, are compatible enough to allow total communicative success. That is, the agents always generate sentences that are correctly understood by the other agents.

All the agents have recursive rules for expressing formulas constructed with unary and binary connectives. Agents a2 and a5 have invented a syntactic category for unary connectives (see table 1). The other agents have specific rules for formulas constructed using negation, which use the same word 'f' preferred by the former agents for expressing negation. The grammar rules used for expressing negation place the word associated with the connective in the second position of the sentence. This is indicated by the number that appears in first place on the right hand side of a grammar rule. For example agent a1 would use the sentence 'ywf' to express the formula $\neg u$, assuming it associates the word 'yw' with the propositional constant u.

Thus the number 1 indicates that the expression associated with the connective is located in the first position of the sentence, the number 2 that it is located in the second position of the sentence, and the number 3 that it is located in the third position of the sentence. We use this convention in order to be able to represent two different types of grammar rules for expressing formulas constructed

Table 1. Grammars constructed by the agents in a particular simulation run

Grammar a1
s([not, Y], R) → 2, f, s(Y,Q), {R is Q · 1}
s([and, Y, Z], T) → 3, dyp, s(Z,Q), s(Y,R), {T is Q · R · 1}
s([X, Y, Z], T) → 3, c3(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c3(or, X) → yi, {X is 1}
c3(iff, X) → iaj, {X is 1}
s([X, Y, Z], T) → 1, c1(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c1(if, X) → bqi, {X is 1}
Grammar a2
s([X,Y],R) → 2, c1(X,P), s(Y,Q), {R is P · Q · 1}
c1(not, X) → f, {X is 1}
s([X, Y, Z], T) → 3, c2(X, P), s(Z,Q), s(Y,R), {T is P · Q · R · 1}
c2(and, X) → dyp, {X is 1}
c2(or, X) → yi, {X is 1}
c2(iff, X) → iaj, {X is 1}
s([X, Y, Z], T) → 1, c3(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c3(if, X) → bqi, {X is 1}
Grammar a3
s([not, Y], R) → 2, f, s(Y,Q), {R is Q · 1}
s([X, Y, Z], T) → 3, c1(X, P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c1(and, X) → dyp, {X is 1}
c1(or, X) → yi, {X is 1}
c1(iff, X) → iaj, {X is 1}
s([X, Y, Z], T) → 1, c2(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c2(if, X) → bqi, {X is 1}
Grammar a4
s([not, Y], R) → 2, f, s(Y,Q), {R is Q · 1}
s([X, Y, Z], T) → 3, c4(X, P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c4(and, X) → dyp, {X is 1}
s([X, Y, Z], T) → 3, c7(X, P), s(Z,R), s(Y,Q), {T is P · R · Q · 1}
c7(or, X) → yi, {X is 1}
c7(iff, X) → iaj, {X is 1}
s([if, Y, Z], T) → 1, bqi, s(Y,Q), s(Z,R), {T is Q · R · 1}
Grammar a5
s([X,Y],R) → 2, c1(X,P), s(Y,Q), {R is P · Q · 1}
c1(not, X) → f, {X is 1}
s([X, Y, Z], T) → 3, c4(X, P), s(Z,Q), s(Y,R), {T is P · Q · R · 1}
c4(and, X) → dyp, {X is 1}
c4(or, X) → yi, {X is 1}
s([X, Y, Z], T) → 3, c2(X, P), s(Y,Q), s(Z,R), {T is P · Q · R · 1}
c2(iff, X) → iaj, {X is 1}
s([if, Y, Z], T) → 1, bqi, s(Y,Q), s(Z,R), {T is Q · R · 1}

using unary connectives (which place the expression associated with the connective in the first and the second position of the sentence, respectively) and six different types of grammar rules for expressing formulas constructed using

binary connectives[6]. This is so because a grammar rule for expressing formulas constructed using binary connectives must specify the position of the expression associated with the connective in the sentence, and the relative positions of the expressions associated with the arguments of the connective in the sentence.

Consider the second and fourth grammar rules of agent a4. Both rules place the expression associated with the connective in the third position of the sentence, but differ in the positions in which they place the expressions associated with the arguments of the connective.

$$s([X,Y,Z],T) \rightarrow 3, c4(X,P), s(Y,Q), s(Z,R), \{T\ is\ P \cdot Q \cdot R \cdot 1\}$$
$$s([X,Y,Z],T) \rightarrow 3, c7(X,P), s(Z,R), s(Y,Q), \{T\ is\ P \cdot R \cdot Q \cdot 1\}$$

The second rule places the expression associated with the first argument of the connective (variable Y) in the first position of the sentence, the expression associated with the second argument (variable Z) in the second position, and the expression associated with the connective in the third position. The fourth rule places the expression associated with the second argument of the connective (variable Z) in the first position in the sentence, the expression associated with the first argument (variable Y) in the second position, and the expression associated with the connective in the third position. Observe the order in which the non-terminals s(Y,Q) and s(Z,R) appear on the right hand sides of both rules.

When analyzing the grammar rules built by the agents we distinguish between commutative and non-commutative binary connectives. Because in order to communicate formulas constructed with commutative connectives, the agents only have to agree on a common vocabulary and on the position in which they place the expression associated with the connective in the sentence.

Consider the case in which two agents agree on a common vocabulary and on the position in which they place the expression associated with a commutative connective in a sentence. Even if both agents differ in the positions in which they place the expressions associated with the arguments of such a connective in the sentence, they will always generate correct interpretations of the sentences generated by each other. Because the difference on the positions of the expressions associated with the arguments of the connective in the sentence can only generate a formula which uses the same connective and which inverts the order of the arguments of such a connective with respect to the formula intended by the speaker. But such a formula will be logically equivalent to the one intended by the speaker, because we are assuming that it is constructed using a commutative connective.

We can observe in table 1 that in fact all agents place in the same position of the sentence (third) the expressions associated with the connectives 'and', 'or' and 'iff', and that they use the same words ('dyp', 'yi' and 'iaj', respectively) for expressing them. But that they do not place in the same positions the expressions associated with the arguments of commutative connectives.

[6] The induction rules (simplification and chunk) have been extended appropriately to deal with this convention.

Table 2. Grammar rules constructed by every agent for expressing disjunctions (i.e., formulas of the form $Y \vee Z$)

Grammar rules constructed for expressing disjunctions: $Y \vee Z$	
Agent a1	s([X, **Y**, Z], T) → 3, c3(X,P), **s(Y,Q)**, s(Z,R), {T is P · Q · R · 1} c3(or, X) → yi, {X is 1}
Agent a2	s([X, **Y**, Z], T) → 3, c2(X,P), s(Z,Q), **s(Y,R)**, {T is P · Q · R · 1} c2(or, X) → yi, {X is 1}
Agent a3	s([X, **Y**, Z], T) → 3, c1(X,P), **s(Y,Q)**, s(Z,R), {T is P · Q · R · 1} c1(or, X) → yi, {X is 1}
Agent a4	s([X, **Y**, Z], T) → 3, c7(X,P), s(Z,R), **s(Y,Q)**, {T is P · R · Q · 1} c7(or, X) → yi, {X is 1}
Agent a5	s([X, **Y**, Z], T) → 3, c4(X,P), s(Z,Q), **s(Y,R)**, {T is P · Q · R · 1} c4(or, X) → yi, {X is 1}

Table 3. Grammar rules constructed by every agent for expressing implications (i.e., formulas of the form $Y \rightarrow Z$)

Grammar rules constructed for expressing implications: $Y \rightarrow Z$	
Agent a1	s([X, Y, Z], T) → 1, c1(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1} c1(if, X) → bqi, {X is 1}
Agent a2	s([X, Y, Z], T) → 1, c3(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1} c3(if, X) → bqi, {X is 1}
Agent a3	s([X, Y, Z], T) → 1, c2(X,P), s(Y,Q), s(Z,R), {T is P · Q · R · 1} c2(if, X) → bqi, {X is 1}
Agent a4	s([if, Y, Z], T) → 1, bqi, s(Y,Q), s(Z,R), {T is Q · R · 1}
Agent a5	s([if, Y, Z], T) → 1, bqi, s(Y,Q), s(Z,R), {T is Q · R · 1}

For example, agents a1 and a3 place the expression associated with the first argument of the connective *or* in the first position of the sentence (see table 2), while agents a2, a4 and a5 place it in the second position of the sentence.

The positions in which the expressions associated with the arguments of non-commutative connectives are placed in a sentence determine however the meaning of the sentence. If the positions of the arguments of a connective in a formula are inverted during the process of interpretation, the formula interpreted by the hearer will not be logically equivalent to the formula intended by the speaker. Differences in the positions of the expressions associated with the arguments of

non-commutative connectives prevent therefore correct communication, and are thus eliminated by the self-organisation process.

We can observe that all agents use the word 'bqi' for expressing the connective 'if', that they all place it in the first position of the sentence, and that all of them place the expressions associated with the antecedent and the consequent of an implication in the same positions of the sentence (see table 3).

All agents have created syntactic categories for commutative connectives, although the extent of such categories differs from one agent to another depending on the positions in which they place the expressions associated with the arguments of the connectives 'and', 'or' and 'iff' in the sentence. Agents a1, a2 and a3 have created syntactic categories for non-commutative connectives, whereas agents a4 and a5 have specific grammar rules for expressing implications.

There are no alternative words for any connective in the agents' grammars, because the mechanism for forgetting rules that have not been useful in past language games has removed such words from the grammars.

We can conclude then that the self-organisation process minimizes the learning effort of the agents by imposing only those order relations among the components of a sentence that are necessary for language understanding.

Figure 1 shows the evolution of the communicative success, averaged over ten simulation runs with different initial random seeds, for a population of five agents. The *communicative success* is the average of successful language games in the last ten language games played by the agents. We can observe that the

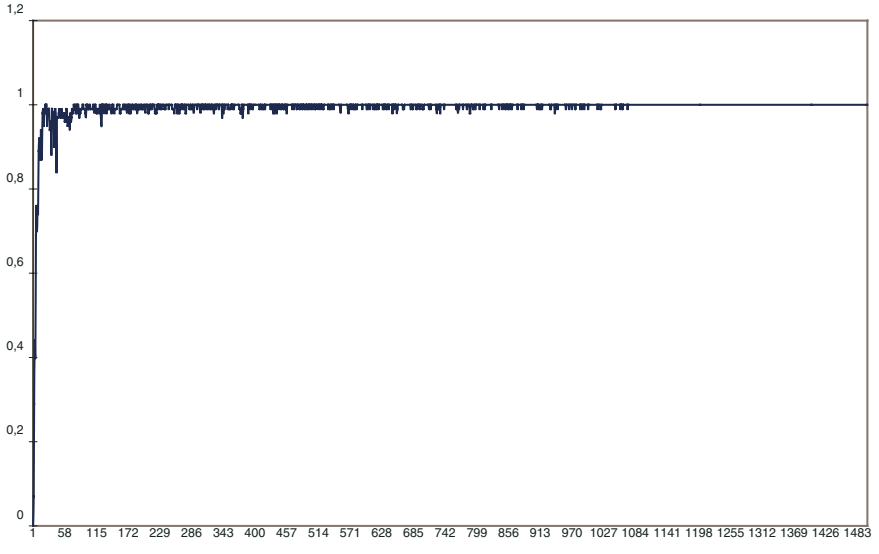


Fig. 1. Evolution of the communicative success in experiments performed using a population of five agents, 10010 language games about propositional constants (not shown), and 15010 language games about formulas constructed using logical connectives

agents reach a communicative success of 100% in 10800 language games. That is, after each agent has played on average 2160 language games about logical formulas and 2000 games about propositional constants.

5 Related Work

The emergence of recursive communication systems in populations of autonomous agents which have no prior linguistic knowledge has been studied by other authors [14,18,21]. The research presented in [18] addresses the problem of the emergence of recursive communication systems in populations of autonomous agents, as we do. It differs from the work described in the present paper by focusing on learning exemplars rather than grammar rules. These exemplars have costs, as our grammar rules do, and their costs are reinforced and discouraged using self-organization principles as well. The main challenge for the agents in the experiments described in [18] is to construct a communication system that is capable of naming atomic formulas and, more importantly, marking the equality relations among the arguments of the different atomic formulas that constitute the meaning of a given sentence. This task is quite different from the learning task proposed in this paper which focusses on categorizing propositional sentences and connectives, and marking the scope of each connective using the order of the constituents of a sentence.

The most important difference between our work and that presented in [14] is that the latter focusses on language transmission over generations. Rather than studying the emergence of recursive communication systems in a single generation of agents, as we do, it shows that the bottleneck established by language transmission over several generations favors the propagation of compositional and recursive rules because of their compactness and generality. In the experiments described in [14] the population consists of a single agent of a generation that acts as a teacher and another agent of the following generation that acts as a learner. There is no negotiation process involved, because the learner never has the opportunity to act as a speaker in a single iteration. We consider however populations of five agents which can act both as speakers and hearers during the simulations. Having more than two agents ensures that the interaction histories of the agents are different from each other, in such a way that they have to negotiate in order to reach agreements on how to name and order the constituents of a sentence.

The induction mechanisms used in the present paper are based on the rules for chunk and simplification in [14], although we have extended them so that they can be applied to grammar rules which have scores and which mark with a number the position of the connective in the sentence. Finally the meaning space used in [14] (a restricted form of atomic formulas of second order logic) is different from the meaning space considered in the present paper (arbitrary formulas from a propositional logic language), although both of them require the use of recursion.

6 Conclusions

We have described some experiments which show how a *language* expressive enough to allow the communication of meanings of the same complexity as propositional logic formulas can emerge in a population of autonomous agents which have no prior linguistic knowledge. This language although simple has interesting properties found in natural languages, such as recursion, syntactic categories for propositional sentences and connectives, and partial word order for marking the scope of each connective.

An approach based on *general purpose cognitive capacities,* such as invention, adoption and induction, and on *self-organisation* principles applied to a particular type of linguistic interaction known as a *language game* has been taken.

These experiments extend previous work by considering a larger population and a much larger search space of grammar rules. In particular the agents are allowed to order the expressions associated with the constituents of a logical formula in arbitrary order in the sentence. Previous work assumed that the expressions associated with the connectives should be always placed in the first position of the sentence. The branching factor of the search space of grammar rules that can be used for expressing formulas constructed with binary connectives has been extended thus from two to six.

Another difference is that communication is considered successful in a language game if the meaning interpreted by the hearer is *logically equivalent* to the meaning the speaker had in mind. In previous experiments [1] both meanings were required to be syntactically equal, i.e., the same formula. This has allowed us to observe how a less strict grammar in terms of word order emerges through the self-organisation process, which minimizes the learning effort of the agents by imposing only those order relations among the components of a sentence that are necessary for language understanding.

In particular the grammar rules built by different agents for communicating formulas constructed using commutative connectives (see table 1) agree on the expression used to refer to each connective and on the position in which the expression associated with each connective is placed in the sentence, but differ on the positions in which the expressions associated with the arguments of the connectives are placed in the sentence. All agents use the expressions `dyp, yi` and `iaj` for referring to the logical connectives *and, or* and *iff*, respectively. The expressions associated with these connectives are placed in the third position in the sentence by all agents as well. But the expression associated with the first argument of these three connectives is placed in the different positions of the sentence by agents a3 and a2. This difference in the order of some constituents of a sentence has not been eliminated by the self-organisation process, because it is perfectly compatible with correct understanding.

Acknowledgments

This work is partially funded by the DGICYT TIN2005-08832-C03-03 project (MOISES-BAR).

References

1. Sierra, J.: Propositional logic syntax acquisition. In: Vogt, P., Sugita, Y., Tuci, E., Nehaniv, C.L. (eds.) Symbol Grounding and Beyond. LNCS (LNAI), vol. 4211, pp. 128–142. Springer, Heidelberg (2006)
2. McCarthy, J.: Formalizing Common Sense. Papers by John McCarthy. Ablex. Edited by Vladimir Lifschitz (1990)
3. Piaget, J.: The Equilibration of Cognitive Structures: the Central Problem of Intellectual Development. University of Chicago Press, Chicago (1985)
4. Santibáñez, J.: Relación del rendimiento escolar en las áreas de lectura y escritura con las aptitudes mentales y el desarrollo visomotor. Universidad Nacional de Educación a Distancia, D.L., Madrid (1984) ISBN 84-398-2486-6
5. Santibáñez, J.: Variables psicopedagógicas relacionadas con el rendimiento en E.G.B. Instituto de Estudios Riojanos, ISBN 84-87252-00-1, Logroño (1988)
6. Santibáñez, J.: La evaluación de la escritura: test de escritura para el ciclo inicial, T.E.C.I. CEPE, D.L. Madrid (1989) ISBN 84-86235-87-1
7. Sierra, J.: Grounded models as a basis for intuitive reasoning. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 401–406 (2001)
8. Sierra, J.: Grounded models as a basis for intuitive and deductive reasoning: The acquisition of logical categories. In: Proceedings of the European Conference on Artificial Intelligence, pp. 93–97 (2002)
9. Sierra, J.: Grounded models as a basis for intuitive reasoning: the origins of logical categories. In: Papers from AAAI–2001 Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems. Technical Report FS-01-01, pp. 101–108. AAAI Press, Menlo Park (2001)
10. Steels, L.: The Talking Heads Experiment. Words and Meanings, vol. 1. LABORATORIUM, Antwerpen (1999) (Special Pre-edition)
11. Steels, L., Kaplan, F., McIntyre, A., V Looveren, J.: Crucial factors in the origins of word-meaning. In: The Transition to Language, pp. 252–271. Oxford University Press, Oxford (2002)
12. Steels, L.: The origins of syntax in visually grounded robotic agents. Artificial Intelligence 103(1-2), 133–156 (1998)
13. Steels, L.: The emergence of grammar in communicating autonomous robotic agents. In: Proceedings of the European Conference on Artificial Intelligence, pp. 764–769. IOS Publishing, Amsterdam (2000)
14. Kirby, S.: Learning, bottlenecks and the evolution of recursive syntax. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models, pp. 96–109. Cambridge University Press, Cambridge (2002)
15. Stolcke, A.: Bayesian Learning of Probabilistic Language Models. PhD thesis, Univ. of California at Berkeley (1994)
16. Vogt, P.: The emergence of compositional structures in perceptually grounded language games. Artificial Intelligence 167(1-2), 206–242 (2005)
17. Steels, L.: The synthetic modeling of language origins. Evolution of Communication 1(1), 1–35 (1997)
18. Batali, J.: The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models, pp. 111–172. Cambridge U.P, Cambridge (2002)

19. Steels, L.: Constructivist development of grounded construction grammars. In: Proc. Annual Meeting of Association for Computational Linguistics, pp. 9–16 (2004)
20. Steels, L., Wellens, P.: How grammar emerges to dampen combinatorial search in parsing. In: Vogt, P., Sugita, Y., Tuci, E., Nehaniv, C.L. (eds.) Symbol Grounding and Beyond. LNCS (LNAI), vol. 4211, pp. 76–88. Springer, Heidelberg (2006)
21. Hurford, J.: Social transmission favors linguistic generalization. In: The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form, pp. 324–352. Cambridge University Press, Cambridge (2000)

Contextualizing Behavioural Substitutability and Refinement of Role Components in MAS

Nabil Hameurlain

LIUPPA Laboratory, Avenue de l'Université, BP 1155, 64013 Pau, France
nabil.hameurlain@univ-pau.fr
http://www.univ-pau.fr/~hameur

Abstract. In this paper we focus on a new approach for the definition of context-based behavioural substitutability and refinement of roles in MAS. First, we introduce two flexible roles compatibility relations depending on the context of use (environment). Then, our formal framework is enhanced with the definition of two flexible behavioral subtyping relations related to the principle of substitutability. We show the existing link between compatibility and substitutability, and namely the preservation of the proposed compatibility relations by substitutability. Finally, we study the action as well as the state-based refinements of roles and investigate the links between the substitutability and the refinements of roles. We show that the proposed behavioural substitutability relations are preserved under the roles refinement.

1 Introduction

Roles are basic buildings blocks for defining the behavior of agents and the requirements on their interactions. Modeling interactions by roles allows a separation of concerns by distinguishing the agent-level and system-level concerns with regard to interactions. Usually, it is valuable to reuse roles previously defined for similar applications, especially when the structure of interaction is complex. To this end, roles must be specified in an appropriate way, since the composition of independently developed roles can lead to the emergence of unexpected interaction among the agents.

Although the concept of role has been exploited in several approaches [2, 3, 14] in the development of agent-based applications, no consensus has been reached about what is a role and namely how it should be specified and implemented. In our previous work [5], we have shown that the facilities brought by the Component Based Development (CBD) approach [13] fit well the issues raised by the use of roles in MAS, and defined RICO (Role-based Interactions COmponents) model for specifying complex interactions based on the roles. RICO proposes a specific definition of role, which is not in contrast with the approaches mentioned above, but is quite simple and can be exploited in specifications and implementations. In the RICO model, when an agent intends to take a role, it creates a new component (i.e. an instance of the component type corresponding to this role) and this role-component is linked to its base-agent. Then, the role is enacted by the role-component and it interacts with the role-components of the other agents.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 193–207, 2008.

In [6], a Petri-net based formal specification for RICO is given together with their compatibility and substitutability semantics. In this paper, we focus on a new approach to the definition of more flexible roles behavioural substitutability/refinement relations depending on the context of use (environment). The proposed approach uses the concept of *usability* of roles and then provides a formal framework for modeling usable role-components and their composition. This paper extends the work presented in [7] and the contributions are: (1) to provide sufficient conditions for deducing emergent properties by role's composition. These sufficient conditions are defined upon more flexible roles compatibility relations, (2) to show the existing link between compatibility and substitutability of roles, and namely the preservation of the proposed compatibility relations by substitutability, (3) to study the (action and state-based) refinement of roles and investigate the compatibility of the substitutability principle with the refinement, which seems to be necessary when we deal with incremental design of role-based complex interactions.

The structure of the paper is as follows. Section 2 presents our Role-based Interactions COmponents (RICO) specification model which is based on the Components-nets formalism that combines Petri nets with the component-based approach. In section 3 we introduce the notion of role's usability and based on that we provide two flexible compatibility relations for roles taking into account the property preservation such as the completion and the proper termination of roles. In section 4, according to the principle of substitutability [11], we study two new behavioural subtyping relations between roles and show that the proposed roles compatibility relations are preserved by substitutability. In this section we also address the consistency of the proposed substitutability relations w.r.t the action and state-based refinements. In section 5 we present conclusion and related approaches.

2 Role-Based Interaction Components Modeling

2.1 The Component-Nets Formalism (C-Nets)

Backgrounds on Labelled Petri nets. A marked Petri net $N = (P, T, W, M_N)$ consists of a finite set P of places, a finite set T of transitions where $P \cap T = \emptyset$, a weighting function $W : P \times T \cup T \times P \to \mathbb{N}$, and $M_N : P \longrightarrow \mathbb{N}$ is an initial marking. The preset of a node $x \in P \cup T$ is defined as ${}^\bullet x = \{y \in P \cup T, W(y, x) \neq 0\}$, and the postset of $x \in P \cup T$ is defined as $x^\bullet = \{y \in P \cup T, W(x, y) \neq 0\}$. Let $M: P \longrightarrow \mathbb{N}$ be a marking of the Petri net. A transition $t \in T$ is enabled under a marking M, noted $M[t>$, if $W(p, t) \leq M(p)$, for each place p. In this case t may occur, and its occurrence yields the follower marking M', where $M'(p) = M(p) - W(p, t) + W(t, p)$, noted $M[t> M'$. The enabling of a sequence of transitions $\sigma \in T^*$ and its occurrence are defined inductively, noted for simplicity $M[\sigma> M'$. We denote as $LN = (P, T, W, M_N, l)$ the (marked, labelled) Petri net in which the events represent actions, which can be observable. It consists of a marked Petri net $N = (P, T, W, M_N)$ with a labelling function $l: T \longrightarrow A \cup \{\lambda\}$. Let ε be the empty sequence of transitions, l is extended to an homomorphism $l^*: T^* \longrightarrow A^* \cup \{\lambda\}$ in the following way: $l(\varepsilon) = \lambda$ where ε is the empty string of T^*, and $l^*(\sigma.t) = l^*(\sigma)$ if $l(t) \in \{\lambda\}$, $l^*(\sigma.t) = l^*(\sigma).l(t)$ if $l(t) \notin \{\lambda\}$. In the following, we denote l^* by l, LN by (N, l), and if $LN = (P, T, W, M_N, l)$ is a Petri

net and l' is another labelling function of N, (N, l') denotes the Petri net (P, T, W, M_N, l'), that is N provided with the labelling l'. A sequence of actions $w \in A^* \cup \{\lambda\}$ is enabled under the marking M and its occurrence yields a marking M', noted M[w>> M', iff either M = M' and $w = \lambda$ or there exists some sequence $\sigma \in T^*$ such that $l(\sigma) = w$ and $M[\sigma> M'$. The first condition accounts for the fact that λ is the label image of the empty sequence of transitions. For a marking M, Reach (N, M) = {M'; $\exists\ \sigma \in T^*$; $M[\sigma> M'$} is the set of reachable markings of the net N from the marking M.

Components nets (C-nets). The Component-nets formalism [6] combines Petri nets with the component-based approach. Semantically, a Component-net involves two special places: the first one is the input place for instance creation of the component, and the second one is the output place for instance completion of the component. A C-net (as a server) makes some services available to the nets and is capable of rendering these services. Each offered service is associated to one or several transitions, which may be requested by C-nets, and the service is available when one of these transitions, called *accept-transitions*, is enabled. On the other hand it can request (as a client) services from other C-net transitions, called *request-transitions*, and needs these requests to be fulfilled. These requirements allow focusing either upon the server side of a C-net or its client side.

Definition 2.1 (C-net)
Let CN = (P ∪ {I, O}, T, W, M_N, l_{Prov}, l_{Req}) be a labelled Petri net. CN is a *Component-net* (C-net) if and only if:

1. The labelling of transitions consists of two labelling functions l_{Prov} and l_{Req}, such that: $l_{Prov} : T \longrightarrow Prov \cup \{\lambda\}$, where $Prov \subseteq A$ is the set of provided services, and $l_{Req} : T \longrightarrow Req \cup \{\lambda\}$, where $Req \subseteq A$ is the set of required services.
2. The set of places contains a specific *Input* place I, such that ${}^\bullet I = \varnothing$ (*Instance creation*),
3. The set of places contains a specific *Output* place O, such that $O^\bullet = \varnothing$ (*Instance completion).*

Notation. We denote by [I] and [O], which are considered as bags, the markings of the Input and the Output place of CN, and by Reach (CN, [I]), the set of reachable markings of the component-net CN obtained from its initial marking M_N within one token in its Input place I. Besides, when we deal with the graphical representation of the C-nets, we use ! and ? keywords for the usual sending (required) and receiving (provided) services together with the labeling function l instead of the two labeling functions l_{Prov} and l_{Req}.

Definition 2.2 (soundness)
Let CN = (P ∪ {I, O}, T, W, M_N, l) be a *Component-net* (C-net). CN is said to be *sound* iff the following conditions are satisfied:

1. Completion option: $\forall\ M \in$ Reach(CN, [I]), $[O] \in$ Reach(CN, M).
2. Reliability option: $\forall\ M \in$ Reach(CN, [I]), $M \geq [O]$ implies M = [O].

Completion option states that, if starting from the initial state, i.e. activation of the C-net, it is always possible to reach the marking with one token in the output place O.

Reliability option states that the moment a token is put in the output place O corresponds to the *termination* of a C-net without leaving dangling references.

Refinement of C-nets. We are interested in place and action (service) refinements of C-nets. Place-refinement consists in replacing a place of the C-net by another C-net. Then the Input (resp. Output) place has exactly the same output (resp. input) transitions as the refined place in the refined net. Whereas the action-based refinement consists of replacing an action of the C-net by another C-net. Then, the output (resp. input) transition of the *Input* (resp. Output) place may have more than one arc from (resp. to) the abstract C-net. These two (canonical) refinements ensure that the output transitions of the Input place fire before the input transitions of the *Output* place in the refined C-net. Our proposals differ from those found in the context of (high level) Petri nets [10] where the substitution of transitions or places, for instance, is like macros or textual substitution. There, they maintain structural compatibility but there is no concept of abstract behaviour. Therefore our proposals are more constrained than textual substitution, since they require behavioural consistency between a refinement and its corresponding abstraction.

Definition 2.3 (place and action -refinement of C-nets)
Let CN = (P ∪ {I, O}, T, W, M_N, l), $a \in$ A be an abstract-action of CN, and CN' = (P' ∪ {I', O'}, T', W', M'_N, l'). Let $\underline{CN'}$ = (P', T', W', M'_N, l') be the component CN' without its Input and Output place.

1. The (place-) refinement of a place p ∈ P by CN' in CN, noted [p/ CN'] CN, is the Component-net [p/ CN'] CN obtained by the substitution of the place p by CN' in the component-net CN. We say that [p/ CN'] CN is a *place-refinement* of CN.
2. The (action-) refinement of an action *a* by CN' in CN, noted [a/ CN'] CN, is the Component-net [a/ CN'] CN obtained by the substitution of all the transitions t such l(t) = a, by $\underline{CN'}$ in the component-net CN. We say that [a/ CN'] CN is an *action-refinement* of CN.

Asynchronous (parallel) composition of C-nets. The *parallel composition* of C-nets, noted ⊕ : C-net × C-net ⟶ C-net, is made by communication places allowing interaction through observable services in an *asynchronous* way. Given a client C-net and a server C-net, it consists in connecting, through the communication places, the request and the accept transitions having the same service names: for each service name, we add one *communication-place* for receiving the requests/replies of this service. Then, all the accept-transitions labeled with the same service name are provided with the same *communication-place*, and the client C-net is connected with the server C-net through these communication places by an arc from each request-transition towards the suitable communication-place and an arc from the suitable communication-place towards each accept-transition. In order to achieve a syntactically correct compound C-net C = A ⊕ B, it is necessary to add new components for initialization and termination: two new places (an Input and Output place), noted {Ic, Oc}, and two new not observable transitions, noted {ti, to}, for interconnecting the input place {Ic} to the original two input places via the first new transition {ti}, and the two original output places to the output place {Oc} via the second new transition

{to}. Thus, the composition of two C-nets is also a C-net, and this composition is commutative and associative.

2.2 Specification of Roles Components and Their Composition/Refinement

In our RICO model [5], a role component is considered as a component providing a set of interface elements (either attributes or operations, which are provided or required features necessary to accomplish the role's tasks), a behavior (interface elements semantics), and properties (proved to be satisfied by the behavior). In this paper, since we are interested in behavioural compatibility and substitutability of roles, when we deal with role components, we will consider only theirs behavioural interfaces; that is the set of (provided and required) services together with the behaviours.

Definition 2.4 (Role Component)
A Role Component (for simplicity) for a role $\Re$, noted RC, is a 2-tuple RC = (Behav, Serv), where,

- Behav is a C-net describing the life-cycle of the role $\Re$.
- Serv is an interface, a set of public elements, through which RC interacts with other role components, for instance messaging interface. It is a pair (Req, Prov), where Req is a set of required services, and Prov is the set of provided services by RC, and more precisely by Behav.

Since the life-cycle of roles is specified by C-nets, we say that a component role satisfies the completion (resp. terminates successfully) if and only if its behaviour that is its underlying C-net satisfies the completion (resp. terminates successfully).

Definition 2.5 (Role Components composition)
A Role (Component), RC = (Behav, Serv), can be composed from a set of (primitive) Role-Components, $RC_i = (Behav_i, , Serv_i)$, i = 1, …, n, noted $RC = RC_1 \otimes \ldots \otimes RC_n$, as follows:

- $Behav = Behav_1 \oplus \ldots \oplus Behav_n$.
- Serv = (Req, Prov) such that $Req = \cup Req_i$, and $Prov = \cup Prov_i$, i=1, …, n.

The composition of two role-components is also a role-component, and this composition is commutative and associative. Besides, we note that the composition operator $\otimes$ allows composition of many instances of a same role, for instance, one instance of the buyer and many instances of the sellers in an auction protocol [5, 6].

Property 2.1 (Role's composition and property preservation)
Let $RC_1 = (Behav_1, Serv_1)$, and $RC_2 = (Behav_2, Serv_2)$, be two role components.
$RC = RC_1 \otimes RC_2$ satisfies the completion option

$$\Rightarrow RC_i,\ i=1,2 \text{ satisfies the completion option.}$$

Example 1. Let's take the example of the ticket service and the customer. Figure 1 shows RC_1 representing the behaviour of the customer, and RC_2 the behaviour of the Ticket-service. The Ticket service initiates the communication by sending one `Ticket`

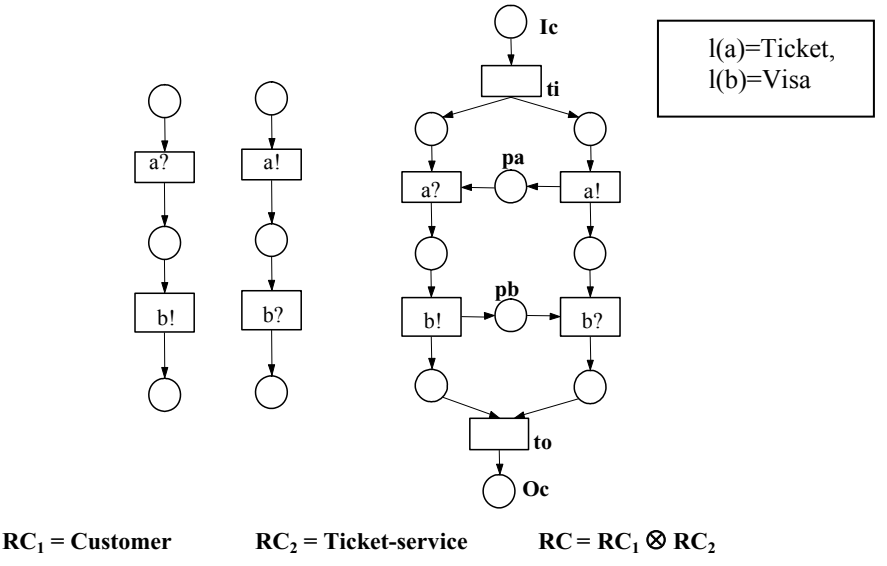


Fig. 1. $RC_1 = (Behav_1, Serv_1)$, $RC_2 = (Behav_2, Serv_2)$, $RC = RC_1 \otimes RC_2 = (Behav, Serv)$, where l(a)= Ticket, l(b) = Visa[1], $Serv_1 = (\{Visa\}, \{Ticket\})$, $Serv_2 = (\{Ticket\}, \{Visa\})$, and $Serv = (\{Visa, Ticket\}, \{Ticket, Visa\})$

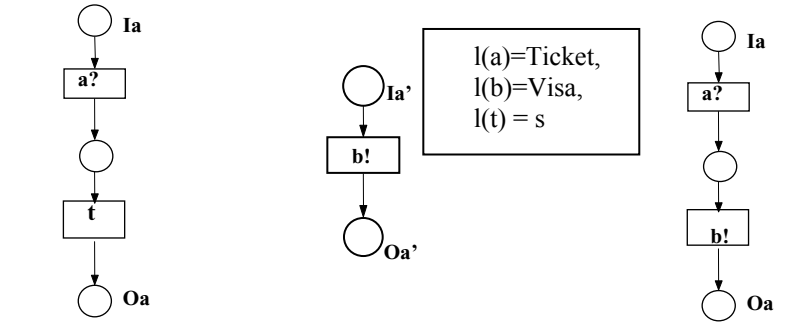
and waits for the payment (`Visa`). By receiving the `Ticket`, the customer makes the payment of the ticket by `Visa`. The result of the composition of the behaviours of RC_1 and RC_2, $RC = RC_1 \otimes RC_2$, is shown in figure 1. These two roles as well as their composed role terminate successfully.

Abstraction/Refinement of roles. Abstraction and refinement are often used together in specification, and validation of large, complex, and open MAS. The abstraction concept is derived from its ability to hide irrelevant details reducing the complexity of interactions, and the refinement is a crucial component in the production of provably reliable software. In the context of open role-based interactions components, the designer tends to start with abstract view of the role, and progressively refines that specification by adding more detail. In this section we use the concept of abstraction/ refinement to design open role based-interactions components. We are interested in both state and action-refinement of roles.

Definition 2.6 (state and action-based Refinement of Roles)
Let $RC_1 = (Behav_1, Serv_1)$ and RC' = (Behav', Serv') be two role components such that $Serv_1 = (Req_1, Prov_1)$ and Serv' = (Req', Prov'). Let p be a place of $Behav_1$ and $s \in Serv_1$ be a service.

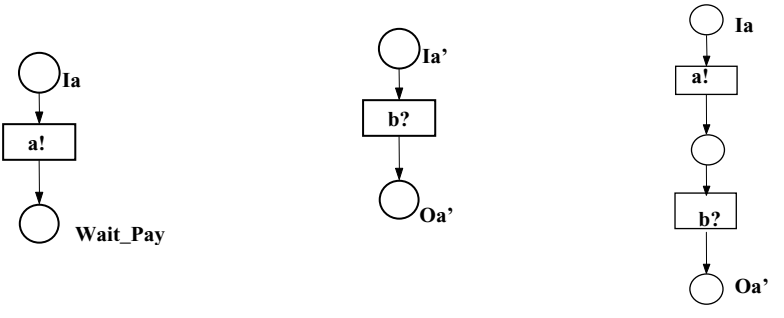
1. The refinement of p by RC' in RC_1 is a role component $RC_2 = (Behav_2, Serv_2)$, noted $RC_2 = [p/ RC'] RC_1$, such that $Behav_2 = [p/ Behav'] Behav_1$ (definition 2.3), and $Serv_2 = Serv_1 \cup Serv'$. We say that RC_2 is a state refinement of RC_1.

[1] The names of transitions are drawn into the box.



RC'$_1$ = Abstract-customer RC$_v$= Visa-payment **RC$_1$= [s/ RC$_v$] RC'$_1$**

Fig. 2. RC$_1$= [s/ RCv] RC'$_1$ is an action refinement of the abstract-customer RC'$_1$



RC'$_2$ =Abstract-T-service RC$_p$ = Visa-Wait-payment RC$_2$ = [Wait_Pay/ RC$_p$] RC'$_2$

Fig. 3. RC$_2$ = [Wait_Pay/ RCp] RC'$_2$ is a place refinement of the abstract-T-service RC'$_2$

2. The refinement of s by RC' in RC$_1$ is a role component RC$_2$ = (Behav$_2$, Serv$_2$), noted RC$_2$ = [s/ RC'] RC$_1$, such that Behav$_2$ = [s/ Behav'] Behav$_1$ (definition 2.3), and Serv$_2$ = (Serv$_1$ \ s) ∪ Serv'. We say that RC$_2$ is an action refinement of RC$_1$.

Example 2. Figure 2 shows RC'$_1$ representing the behaviour of an abstract-customer, and RCv the behaviour of the Visa-payment. The refinement of the service s by the Visa-payment RCv in the abstract-customer RC'$_1$ is the customer component RC$_1$= [s/ RC$_v$] RC'$_1$. Besides, figure 3 shows RC'$_2$ representing the behaviour of an abstract-Ticket-service, and RC$_p$ the behaviour of the Visa-wait-payment. The refinement of the place Wait_pay by the role RCp in the abstract-Ticket-service RC$_2$' is the ticket-service component RC$_2$ = [Wait_pay/ RC$_p$] RC$_9$.

Property 2.2 (Role's refinement and property preservation)
Let RC = (Behav, Serv), and RC' = (Behav', Serv'), be two role components. Let p be a place of Behav and s ∈ Serv be a service.

1. RC and RC' satisfy the completion option (resp. terminate successfully) ⇒
   [p/ RC'] RC satisfies the completion option (resp. terminates successfully).
2. RC and RC' satisfy the completion option (resp. terminate successfully) ⇒
   [s/ RC'] RC satisfies the completion option (resp. terminates successfully).

3 Context-Based Behavioral Compatibility of Role Components

In component-based software engineering, classical approaches for components compatibility deal with components composition together with their property preservation [1]. In our previous work, we have used this approach for role-based interactions components and propose two compatibility relations [6]. The first one deals with the correctness of the composition of roles when reasoning about the completion option, and the second deals with the proper (or successful) termination of the composed role. In this paper, we will consider explicitly the context of use of roles that is their environment in the definition of role-components compatibility relations. The proposed compatibility relations [7], called optimistic compatibility relations, are then more flexible. First, let define the notion of the environment of a role.

Definition 3.1 (Environment)
Let $RC_1 = (Behav_1, Serv_1)$ and $RC_2 = (Behav_2, Serv_2)$, be two roles such that $Serv_i = (Req_i, Prov_i)$, i=1, 2.

RC_2 is called an environment-role (or environment for simplicity) of RC_1, and vice versa, iff $Req_1 = Prov_2$, $Req_2 = Prov_1$.

We let ENV(RC), the set of environments of the role component RC.

Definition 3.1 expresses that for two role components $RC_1 = (Behav_1, Serv_1)$ and $RC_2 = (Behav_2, Serv_2)$, such that both sets of interfaces $Serv_1$ and $Serv_2$ completely match, i.e. $Req_1 = Prov_2$ and $Req_2 = Prov_1$, the role component RC_1 is considered *a role-environment* (or *environment* for simplicity) of RC_2 - and vice versa.

Given a role-component and its environment, it is possible to reason about the completion and the proper termination of their composition. Based on that, we define two notions of usability: the first one, called weak usability, deals with the completion option, and the second one, which is more restrictive than the weak usability, deals with the successful termination property.

Definition 3.2 (Weak and Strong usability)
1. RC is weakly usable iff $\exists$ Env $\in$ ENV(RC), Env $\otimes$ RC satisfies the completion option. We say that Env *weakly utilizes* RC.
2. RC is strongly usable iff $\exists$ Env $\in$ ENV(CR), Env $\otimes$ RC terminates successfully. We say that Env *strongly utilizes* RC.

Example 3. Let take again the example of the ticket service and the customer. Now, the Ticket service initiates the communication by sending (two) `Tickets` and waits of their payment (`VISA` and/or `eCash`). Figure 4 shows RC_3 representing the behaviour of the customer, and RC_4 the behaviour of the Ticket-service. By receiving the `Tickets`, the customer determines the kind of payment of these two tickets. It

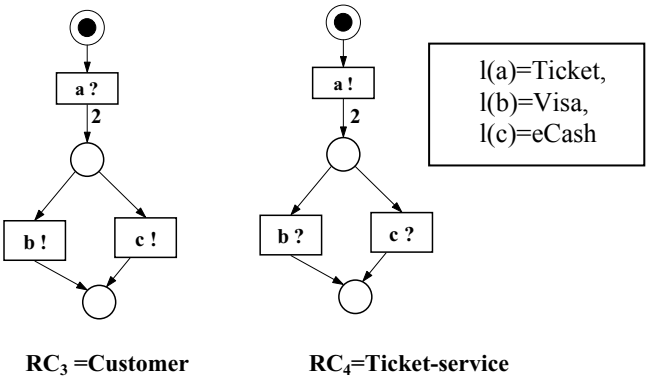


Fig. 4. RC_3 weakly utilizes RC_4, where $Serv_3$ = ({Visa, eCash}, {Ticket}), $Serv_4$ = ({Ticket}, {Visa, eCash})

is easy to prove that roles RC_3 and RC_4 are weakly usable, since RC_3 weakly utilizes RC_4 and vice versa. The role RC_3 is not strongly usable since, its unique (and weakly usable) environment is the role RC_4, and $RC_3 \otimes RC_4$ satisfies the completion option but does not terminate successfully.

Example 4. In figure 5, the ticket service RC_7 initiates the communication by sending one `Ticket` and wait of the payment (either `Visa` or `eCash`). The role components RC_5 and RC_6 are then two examples of the customer's behaviour. By receiving the `Ticket`, they solve an internal conflict and determine the kind of payment. The roles $R5_3$ and RC_7 (resp. RC_6 and RC_7) are strongly usable, since for instance RC_5 strongly utilizes RC_7 (resp. RC_6 strongly utilizes RC_7) and vice versa. Last but not least, let us take the ticket service RC' shown in figure 6. RC' is not weakly usable since there is no environment which can weakly utilize it. Indeed, roles RC_5 and RC_6 are the two possible role-environments of RC' (according to the behaviour of RC' described by the language {`Ticket!.Visa?`, `Ticket!.eCash?`}), nevertheless $RC_5 \otimes$ RC' (as well as $RC_6 \otimes$ RC') yields to a deadlock, since for instance the sequence {`Ticket!.Ticket?.eCash!`} in $RC_5 \otimes$ RC' (as well as in $RC_6 \otimes$ RC') yields to a deadlock- marking, that is a marking where no transition is enabled. This is because of an error in role-component RC': an internal decision (either `Visa?` or `eCash?`) is made, when sending the `Ticket`, and not communicated properly to the environment [1].

We are finally ready to give adequate definitions for roles behavioural optimistic compatibility, which are based on the weak and the strong usability.

Definition 3.3 (Weak and strong optimistic compatibility)
Let RC_1 and RC_2 be two weakly (resp. strongly) usable roles.

RC_1 and RC_2 are Weakly (resp. Strongly) Optimistic Compatible, noted $RC_1 \approx_{WOC} RC_2$ (resp. $RC_1 \approx_{SOC} RC_2$), iff $RC_1 \otimes RC_2$ is weakly (resp. strongly) usable.

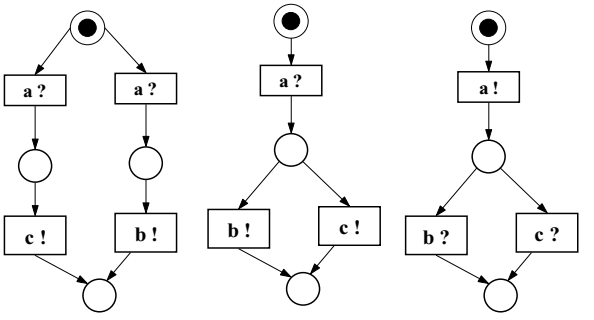


Fig. 5. RC_5 strongly utilizes RC_7, RC_6 strongly utilizes RC_7, where $Serv_5$ = $Serv_6$ = ({Visa, eCash}, {Ticket}) and $Serv_7$ = ({Ticket}, {Visa, eCash})

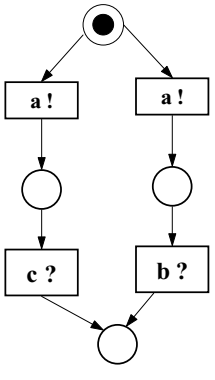


Fig. 6. RC' is not weakly usable, where Serv' = ({Ticket}, {Visa, eCash})

Property 3.1 (Hierarchy of compatibility relations)
Optimistic compatibility relations form a hierarchy: $\approx_{SOC} \Rightarrow \approx_{WOC}$

Example 5. As an example, roles RC_3 and RC_4, shown in figure 6, are weakly (but not strongly) optimistic compatible that is $RC_3 \approx_{WC} RC_4$ holds since $RC_3 \otimes RC_4$ is weakly usable. Indeed, $RC_3 \otimes RC_4$ satisfies the completion option. Besides, the two roles RC_5 and RC_7 shown in figure 5 are strongly optimistic compatible that is $RC_5 \approx_{SOC} RC_7$ holds since $RC_5 \otimes RC_7$ is strongly usable. Indeed, $RC_5 \otimes RC_7$ terminates successfully.

Property 3.2 (Emergent property by Role's composition)
Let RC_1 = ($Behav_1$, $Serv_1$), and RC_2 = ($Behav_2$, $Serv_2$), be two weakly (resp. strongly) compatible role components. Let RC = $RC_1 \otimes RC_2$.

RC_i, i =1,2 satisfies the completion option (resp. terminates successfully) $\Rightarrow \exists$ Env $\in$ ENV(RC), Env $\otimes$ RC satisfies the completion option (resp. terminates successfully).

4 Behavioural Substitutability and Refinements of Role Components

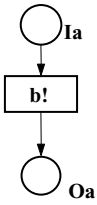
Our main interest is to define behavioural subtyping relations (reflexive and transitive) capturing the principle of substitutability [11] and taking into account the context of use (environment). First, we define two subtyping relations between roles that are based upon the preservation of the (weakly or strongly) utilisation of the former role by any role of its environment. Then we show the existing link between compatibility and substitutability concepts, and namely their combination, which seems necessary, when we deal with incremental design of usable role components. Finally, we investigate the links between substitutability and (action as well as the state-based) refinements of roles.

Definition 4.1 (Weak and Strong optimistic substitutability)
Let $RC_1 = (Behav_1, Serv_1)$, $RC_2 = (Behav_2, Serv_2)$, and $Serv_i = (Req_i, Prov_i)$, i=1, 2, such that: $Prov_1 \subseteq Prov_2$, $Req_1 \subseteq Req_2$.

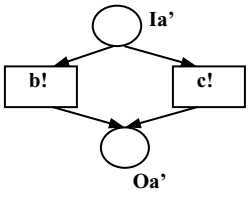
1. RC_2 is less equal to RC_1 w.r.t Weak Substitutability, denoted $RC_2 \leq_{WOS} RC_1$, iff $\forall$ Env $\in$ ENV(RC_1), Env weakly utilizes $RC_1 \Rightarrow$ Env weakly utilizes RC_2.
2. RC_2 is less equal to RC_1 w.r.t Weak Substitutability, denoted $RC_2 \leq_{SOS} RC_1$, iff $\forall$ Env $\in$ ENV(RC_1), Env strongly utilizes $RC_1 \Rightarrow$ Env strongly utilizes RC_2.

The Weak (resp. Strong) Substitutability guarantees the transparency of changes of roles to their environment. Namely, the weak (resp. strong) compatibility between the former role and its environment should not be affected by these changes. In both *weak* and *strong* subtyping relations, the (super-) role component RC_1 can be substituted by a (sub-) role component RC_2 and the environment of the former role RC_1 will not be able to notice the difference since: (a) the sub-role has a larger set of required and provided services ($Req_1 \subseteq Req_2$ and $Prov_1 \subseteq Prov_2$) than the super-role, and (b) any environment that weakly (resp. strongly) utilizes the former role is also able to weakly (resp. strongly) utilize the new role.

Example 6. As an example, consider the roles RC_v and RC'_v given in the figure 7. It is easy to check that $RC'_v \leq_{SOS} RC_v$ since ENV (RC_v) = {RC_p} given in figure 3 and RC_p that strongly utilizes RC_v also strongly utilizes RC'_v. Besides, consider the roles RC_3 and RC_6. $RC_3 \leq_{WOS} RC_6$ holds since the unique environment that strongly (and then weakly) utilizes CR_6 is the role RC_7, and $RC_7 \otimes RC_3$ satisfies the completion option. These two roles RC_3 and RC_6 are not related by the strong subtyping relation that is $RC_3 \leq_{SOS} RC_6$ does not hold, since $RC_3 \otimes RC_7$ does not terminate successfully. Last but not least, consider the roles RC_5 and RC_6; $RC_5 \leq_{SOS} RC_6$ holds since the role RC_7 (which is the unique environment) that strongly utilizes RC_6 also strongly utilizes RC_5. Indeed $RC_5 \otimes RC_7$ terminates successfully.



RC_v= Visa-payment



RC'_v = Visa-eCash-payment

Fig. 7. RC'_v is strongly subtyping RC_v, where Serv = {{Visa}, ∅) and Serv' = ({Visa, eCash}, ∅)

Property 4.1 (Hierarchy of subtyping relations)
Let $\Re C = \{RC_1, \ldots, RC_n\}$ be the set of role components in the system. The relations $\leq_H$, $H \in \{WOS, SOS\}$, are preorder (reflexive and transitive) on $\Re C$ and form a hierarchy: $\leq_{SOS} \Rightarrow \leq_{WOS}$.

The following core theorem of this paper states two fundamental properties of protocols substitutability and their compatibility. First, substitutability is compositional: in order to check if $Env \otimes RC_2 \leq_H Env \otimes RC_1$, $H \in \{WOS, SOS\}$, it suffices to check $RC_2 \leq_H RC_1$, since the latter check involves smaller roles and it is more efficient. Second, substitutability and compatibility are related as follows: we can always substitute a role RC_1 with a sub-role RC_2, provided that RC_1 and RC_2 are connected to the environment Env by the same provided services that is: $Req \cap Prov_2 \subseteq Req \cap Prov_1$. This condition is due to the fact that if the environment utilizes services provided by RC_2 that are not provided by RC_1, then it would be possible that new incompatibilities arise in the processing of these provided services.

Theorem 4.1 (compositionality and compatibility preservation)
Let $RC_1 = (Behav_1, Serv_1)$, $RC_2 = (Behav_2, Serv_2)$ be two roles where $Serv_i = (Req_i, Prov_i)$, i = 1, 2. Let Env = (Behav, Serv) such that $Req \cap Prov_2 \subseteq Req \cap Prov_1$.

1. $Env \approx_{WOC} RC_1$ and $RC_2 \leq_{WOS} RC_1 \Rightarrow Env \approx_{WOC} RC_2$ and $Env \otimes RC_2 \leq_{WOS} Env \otimes RC_1$.
2. $Env \approx_{SOC} RC_1$ and $RC_2 \leq_{SOS} RC_1 \Rightarrow Env \approx_{SOC} RC_2$ and $Env \otimes RC_2 \leq_{SOS} Env \otimes RC_1$.

Example 7. Let take again the roles RC_7, RC_3 and RC_6. We have $RC_7 \approx_{WOC} RC_6$ and $RC_3 \leq_{WOS} RC_6$. Then, according the above theorem, we can deduce that $RC_7 \approx_{WOC} RC_3$ and $RC_7 \otimes RC_3 \leq_{WOS} RC_7 \otimes RC_6$. Besides, we have $RC_7 \approx_{SOC} RC_6$ and $RC_5 \leq_{SOS} RC_6$. According the second part of the above theorem, we can deduce that $RC_7 \approx_{WOC} RC_5$ and $RC_7 \otimes RC_5 \leq_{WOS} RC_7 \otimes RC_6$.

The following two theorems address one key issue of component based software development of roles, *consistency;* that is the compatibility of the substitutability's principle with the role Components (state and action) refinement. First, the weak and strong optimistic substitutability relations are preserved by state refinement of roles.

Second, these two substitutability relations are compositional for the action refinement of roles. These two results are necessary for designing complex and open role-based interactions components in an incremental way, since substitutability (and then compatibility, according to the theorem 4.1) between role components specifications and/ or between roles specifications and their implementations are preserved.

Theorem 4.2 (preservation of substitutability by state refinement)
Let RC = (Behav, Serv) and RC' = (Behav', Serv') be two role components such that Serv = (Req, Prov), Serv' = (Req', Prov') and Serv $\cap$ Serv' = $\varnothing$. Let p be a place of Behav.

1. RC' weakly usable $\Rightarrow$ [p/ RC'] RC $\leq_{WOS}$ RC,
2. RC' strongly usable $\Rightarrow$ [p/ RC'] RC $\leq_{SOS}$ RC.

Theorem 4.3 (preservation of substitutability by action refinement)
Let RC_1, RC_2 be two usable Role Components such that: $Prov_1 \subseteq Prov_2$, $Req_1 \subseteq Req_2$. Let $s \in Prov_1 \cup Req_1$. Let RC'_1, RC'_2 be two usable Role Components such that: $Prov'_1 \subseteq Prov'_2$, $Req'_1 \subseteq Req'_2$, and $Serv_1 \cap Serv'_1 = Serv_2 \cap Serv'_2 = \varnothing$. Then:

1. $RC_2 \leq_{WOS} RC_1$ and $RC'_2 \leq_{WOS} RC'_1 \Rightarrow$ [s/ RC'_2] $RC_2 \leq_{WOS}$ [s/ RC'_1] RC_1,
2. $RC_2 \leq_{SOS} RC_1$ and $RC'_2 \leq_{SOS} RC'_1 \Rightarrow$ [s/ RC'_2] $RC_2 \leq_{SOS}$ [s/ RC'_1] RC_1.

Example 8. As an example, consider for instance the roles RC'_2, RC_p and RC_2 = [Wait_Pay/ RC_p] RC'_2, given in figure 3. We can check that RC_p is strongly (and then weakly) usable, and deduce according to the theorem 4.2 that $RC_2 \leq_{SOS} RC'_2$. Besides let take the roles RC_v and RC'_v given in the figure 7. We have $RC'_v \leq_{SOS} RC_v$, and according to the theorem 4.3, we can deduce that [s/ RC'_v] $RC'_1 = RC_1 \leq_{SOS}$ [s/ RC_v] $RC'_1 = RC_6$, where RC'_1 is the role given in figure 2.

5 Conclusion and Related Work

The aim of this paper is to present a new approach to the definition of context-based behavioural substitutability and refinement for role-components in MAS. The paper provides a framework for modelling usable role-components together with their composition. This framework is discussed in terms of compatibility and substitutability checks. We proposed two new and flexible compatibility relations together with two subtyping relations between role-components taking into account the non-determinism, the composition mechanism of role-components, as well as the property preservation such as the completion and the proper termination of roles. We furthermore investigated some properties related to our substitutability and compatibility relations, namely the preservation of the compatibility by substitutability as well as the compositionality of the proposed substitutability relations. Finally, we studied the behavioural refinement of roles and investigated the existing links between roles substitutability and refinement by showing the consistency of the proposed substitutability relations w.r.t. (state and action) refinement of roles.

Related work. The idea of optimistic approach to the definition of components compatibility has been originally introduced in [1] for interface automata. Unlike traditional uses of automata, the authors proposed an optimistic approach to automata composition. Two interface automata are (optimistic) compatible, if they are composable and there exists a legal environment for these two automata, i.e. an environment such that no deadlock state is reachable in the automata obtained by the composition of the two interface automata and that environment. This work is close to ours, since our weak optimistic compatibility relation for role-components is related to the optimistic compatibility relation defined for automata composition. Besides, our approach can be seen as an extension of this work, since the existing link between the proposed compatibility and subtyping relations is studied. In [8], the concept of usability is used for analyzing web service based business processes. The authors defined the notion of usability of workflow modules, and studied the soundness of a given web service, considering the actual environment it will be used in. Based on this formalism together with the notion of usability, the authors present compatibility and equivalence definitions of web services. This approach is close to ours, since the compatibility of two workflow modules is related to our strong optimistic compatibility of role-components. Furthermore, in that work, the notion of equivalence of web services is defined upon the usability notion, but it is not linked to the compatibility. Whereas in our work, we define in addition the notion of weak optimistic compatibility, which is less restrictive than the strong optimistic compatibility, and the substitutability of role-components was addressed together with the preservation of compatibility by substitutability. In [2], authors define the notion of compatibility and consistency between components of the agents and role: the agent is compatible with a role if the agent (sub) goals are subset of (sub) goals of the role. Conversely a role is compatible with the agent if the (sub) goals of the role are a subset of the agent (sub) goals. This compatibility relation indicates that the agent is highly suitable to fulfil the role. Nevertheless such a match between the agent and roles is not always possible. Then, the authors introduce a weaker relation, called consistency, which indicates that the goals of the agent and the roles do not conflict. This approach is close to the compatibility relations that have been proposed in our previous work [6], since it is related explicitly to the property preservation, and then one of the limitations is that these two relations are very strong. Whereas in the approach presented in this paper, we have proposed more flexible behavioural compatibility relations taking into account the environment in which the agent-roles are involved. Finally, in [4], the authors contextualize commitment protocols by adapting them, via different transformations, on context and the agent's preferences based on that context. A protocol is transformed by composing its specification with a transformer specification, and the contextualization is characterized operationally by relating the original and transformed protocols according to protocol substitutability relations. There, the semantics of protocol substitutability relations are based on state similarity to compare states occurring in runs of protocols. This approach is close to ours, since the proposed protocols transformations are related to our (state) refinement. Nevertheless, in practice, checking state-similarity between protocols is not quite easy, since the runs of protocols are needed for that purpose. In contrast, our approach which is related to the composition and

refinement of roles, like others proposed in [9, 12] for protocols, proposed structural (state and action based-) refinement of role-based interactions components, ensuring correctness guarantees w.r.t their usability; that is the preservation of the utilizing of that roles by their environment.

References

1. De Alfaro, L., Henzinger, T.A.: Interface automata. In: Proc of ESEC/FSE. Software Engineering Notes, vol. 26(5), pp. 109–120. ACM, New York (2001)
2. Dastani, M., Dignum, V., Dignum, F.: Role Assignment in Open Agent Societies. In: AAMAS 2003, pp. 489–495. ACM, New York (2003)
3. Cabri, G., Leonardi, L., Zambonelli, F.: BRAIN: a Framework for Flexible Role-based Interations in Multi-agent Systems. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 145–161. Springer, Heidelberg (2003)
4. Chopra, A.-K., Singh, M.P.: Contextualizing Commitment Protocols. In: 5th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), pp. 1345–1352. ACM, New York (2006)
5. Hameurlain, N., Sibertin-Blanc, C.: Specification of Role-based Interactions Components in MAS. In: Choren, R., Garcia, A., Lucena, C., Romanovsky, A. (eds.) SELMAS 2004. LNCS, vol. 3390, pp. 180–197. Springer, Heidelberg (2005)
6. Hameurlain, N.: Formalizing Compatibility and Substitutability of Role-based Interactions Components in Multi-agent Systems. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 153–162. Springer, Heidelberg (2005)
7. Hameurlain, N.: Formalizing Context-Based Behavioural Compatibility and Substitutability of Roles in MAS. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 153–162. Springer, Heidelberg (2007)
8. Martens, A.: Analyzing Web Service Based Business. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 19–33. Springer, Heidelberg (2005)
9. Mazouzi, H., El Fallah Seghrouchni, A., Haddad, S.: Open Protocol Design for Complex Interactions in MAS. In: AAMAS 2002, pp. 517–526. ACM, New York (2002)
10. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proc. of the IEEE 77(4), 541–580 (1989)
11. Liskov, B.H., Wing, J.M.: A Behavioral Notion of Subtyping. ACM Trans. on Programing Languages and Systems 16(6), 1811–1841 (1994)
12. Vitteau, B., Huget, M.-P.: Modularity in Interaction Protocols. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 291–309. Springer, Heidelberg (2004)
13. Szyperski, C.: Component Software-Beyond Object-Oriented Programming. Addison-Wesley, Reading (2002)
14. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology. ACM TOSEM 12(3), 317–370 (2003)

Amongst First-Class Protocols

Tim Miller[1] and Jarred McGinnis[2]

[1] Department of Computer Science
University of Liverpool, Liverpool, L69 7ZF
tim@csc.liv.ac.uk
[2] Department of Computer Science
Royal Holloway, University of London, Egham, Surrey TW20 0EX
jarred@cs.rhul.ac.uk

Abstract. The ubiquity of our increasingly distributed and complex computing environments have necessitated the development of programming approaches and paradigms that can automatically manage the numerous tasks and processes involved. Hence, research into agency and multi-agent systems are of more and more interest as an automation solution. Coordination becomes a central issue in these environments. The most promising approach is the use of interaction protocols. Interaction protocols specify the interaction or social norms for the participating agents. However the orthodoxy see protocols as rigid specifications that are defined *a priori*. A recent development in this field of research is the specification of protocols that are treated as first-class computational entities. This paper explores the most prominent approaches and compares them.

1 Introduction

Research into multi-agent systems aims to promote autonomy and intelligence into software agents. Intelligent agents should be able to interact socially with other agents, and adapt their behaviour to changing conditions. Despite this, research into interaction in multi-agent systems is focused mainly on the documentation of interaction protocols *a priori*. We identify three significant disadvantages with this approach: 1) it strongly couples agents with the protocols they use — something which is unanimously discouraged in software engineering — therefore requiring agent code be changed with every change in a protocol; 2) agents can only interact using protocols that are known at design time, a restriction that seems out of place with the goals of agents being intelligent and adaptive; and 3) agents cannot compose protocols at runtime to bring about more complex interactions, therefore restricting them to protocols that have been specified by human designers — again, this seems out of place with the goals of agents being intelligent and adaptive. An important corollary of these points is that the protocol is internalised within the individual agents. There is no possibility to communicate, inspect or verify the protocols by the agent or others. Not to mention a repetition of effort for each agent engineer as each agent must be encoded with the same protocol.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 208–223, 2008.

Recent research into multi-agent system protocols has begun to focus on *first-class protocols*, which were first defined in [11]. By first-class, we mean that a protocol exists as a computational entity in a multi-agent system that can be shared between agents, referenced, invoked, and composed, in contrast to hard-coded protocols, which exist merely as abstractions that emerge from the messages sent by the participants. Agents in the system read the definition of a protocol to learn its rules and meanings, and then structure their interaction from this definition.

Authors of such work envisage systems in which agents have access to libraries of protocols. Agents can search through these libraries at runtime to find protocols that best suit the goal they are trying to achieve, and can share these protocol specifications with possible future participants. If no single protocol is suitable for the agent, runtime composition of these may offer an alternative.

This paper explores the current state-of-the-art in first-class protocol languages, and compares these. We look at the following approaches:

1. In Section 3, we explore *commitment machines* [17], a socially-centric approach that uses commitment to define the meaning of messages and protocols. We review three different approaches [4,5,18], all of which share the common feature of using Yolum and Singh's commitment machines framework [17].
2. In Section 4, we explore a normative approach to defining protocols [2], which using obligations, permissions, and prohibitions to specify protocols.
3. In Section 5, we explore the Lightweight Coordination Calculus (LCC), a protocol language described in [13], based on process algebra and logic programming.
4. In Section 6, we explore the $\mathcal{RASA}$ language [11], which combines process algebra and constraint languages.
5. In Section 7, we explore an approach [3] that extends Petri Nets to specify message sequencing and message meaning.

In Section 8, we compare and contrast the above approaches, outlining the relative advantages and strengths of the approaches, and in Section 9, we briefly present some approaches that resemble first-class protocol languages, and discuss why they are not.

2 First-Class Protocols — A Definition

Our notion of first-class protocol is comparable to the notion of first-class object/entity in programming languages [15]. That is, a first-class protocol is a referencable, sharable, manipulable entity that exists as a runtime value in a multi-agent system. From the definition of a first-class protocol, participating agents should be able to inspect the definition to learn the rules and effects of the protocol by knowing only the syntax and semantics of the language, and the ontology used to describe rules and effects.

To this end, we define four properties that constitute a first-class protocol language:

- *Formal:* The language must be formal to eliminate that possibility of ambiguity in the meaning of protocols, to allow agents to reason about them using their machinery, and to allow agents to pass and store the protocol definitions as values.
- *Meaningful:* The meaning of messages must be specified by the protocol, rather than simply specifying arbitrary communication actions whose semantics are defined outside the scope of the document. Otherwise, one may encounter a communicative action of which they do not know the definition, rendering the protocol useless.
- *Inspectable/executable:* Agents must be able to reason about the protocols at runtime in order to derive the rules and meaning of the protocol, so that they can determine the actions that best achieve their goals, and compare the rules and effects of different protocols.
- *Dynamically composable:* If an agent does not have access to a protocol that helps to achieve its goals, then it should be able to compose new protocols that do at runtime, possibly from existing protocols. This new protocol must also form a first-class protocol in its own right.

This definition of first-class protocol eliminates many of the protocol specification languages that have been presented in the literature. We emphasise here that first-class does not equal *global*. By global, we mean languages that specify the protocol from a global view of the interaction, rather than from the view of the individual participants. Therefore, languages such as AgentUML and FSM-based languages are not first-class, as is commonly commented, even though they are global. AgentUML is not meaningful (although one could adapt it quite easily to make it meaningful), and the composability at runtime could also be difficult, if possible at all. FSM approaches could also add meaning, but the authors are not aware of any current FSM approaches that are executable and support dynamic composition.

3 Commitment Machines

Yolum and Singh [17] present commitment machines, which are used to formally represent the social relationships that exists between autonomous agents in the form of commitments. A conditional commitment for debtor a to bring about condition q when p is satisfied is represented using the constraint $CC(a, b, p, q)$, in which b is the creditor of the commitment. Non-conditional commitments (or base-level commitments) are written $C(a, b, p)$, which is equivalent to $CC(a, b, true, p)$. In commitment machines, agents participating in a protocol create and manipulate commitments as a result of sending particular messages. Agents are programmed to understand the meaning of commitments, and can therefore reason about protocols whose meaning is specified using commitment machines.

Winikoff [16] presents a mapping from commitment machines to an abstract programming language for agents called the Simple Abstract Agent Programming Language (SAAPL). This approach is somewhat different to our idea of first-class protocol languages in that the agents do not inspect protocols to decide their course of action, but are instead implemented as a mapping from the commitment machine into a SAAPL program.

Several approaches have used the idea of commitment machines for specifying first-class protocols. In this section, we present these approaches.

3.1 Commitment Machines in the Event Calculus

Yolum and Singh [18] use the Event Calculus for specifying commitment machines. The Event Calculus is a logical language for specifying at which time-points actions occur, and the effect that those actions have. Yolum and Singh's approach uses the Event Calculus to specify message sending as actions, and the effect that message sending has is specified as the creation or manipulation of commitments.

Two predicates in the Event Calculus are the most used for specifying protocols. The $Happens$ predicate specifies that an event, e, happens at the time t, written $Happens(e, t)$. The $HoldsAt$ predicate specifies that a property, p, holds at time t, written $HoldsAt(p, t)$. The set of time points is a partially ordered set, with the relation $<$, therefore, one can specify the occurrence of messages using $Happens$, and order them using $<$. For example,

$$Happens(m_1, t_1) \wedge Happens(m_2, t_2) \wedge t_1 < t_2$$

specifies that the message m_1 occurs before the message m_2. To specify further that the sending of m_2 commits agent a to perform p for agent b, we add to the above predicate, the following:

$$HoldsAt(t_2, C(a, b, p)).$$

Yolum and Singh present a set of axioms relating communicative acts and commitments, discuss the use of an abductive planner for agents to plan their execution paths.

3.2 Commitment Machines in OWL-P

Desai *et al.* present OWL-P [4], an ontology used for modelling protocols — specifically business protocols —, which is encoded in the OWL web ontology language. The ontology defines concepts such as message, protocol, roles, proposition, and commitment. Commitments are specified as discussed above, and therefore, a protocol specified using OWL-P is a commitment machine. An additional ontology is presented for protocol composition — that is, composing protocols that achieve a single business goal into protocols that achieve multiple business goals. The axioms that define the composition must be specified by the protocol designer themselves.

3.3 Commitment Machines in MAD-P

Desai and Singh [5] present MAD-P, an extension of the C+ language. The MAD-P approach is similar to that of the Event Calculus approach discussed in Section 3.1. Message passing is specified as actions occurring at particular time points, and the meaning of message passing is specified using commitments, with the relation **causes** linking particular messages to their meaning, for example,

$$m_1 \textbf{ causes } cancel(C(a, b, p)).$$

Sequencing of messages is specified using the **before** relation, used in the context

$$m_1 \textbf{ before } m_2$$

meaning that message m_1 occurs before message m_2.

Desai and Singh present a set of axioms for composing new protocols from existing protocols.

3.4 Discussion

The approaches outlined in this section are all different ways of specifying commitment machines. We note the following properties of all of these approaches:

- Protocols are specified from a global rather than local perspective.
- Message sequencing is specified in a declarative manner, rather than an algebraic/operational manner.
- The languages used for specifying the message sequences and the meaning of messages are the same. That is, the language, for example, the Event Calculus, is used to specify the order in which messages can occur, as well as the preconditions and effects of messages.
- All of the approaches assume some form of state — mainly the existence of commitments between agents.
- The OWL-P and MAD-P approaches support protocol composition, however, composition axioms are at a different level to protocol specification. The Event Calculus approach presented in [18] does not discuss composition, but it seems likely that composition axioms could be defined.

4 Normative Systems

Artikis *et al.* [2] propose an approach similar to commitment machines, especially the commitment machines based on C+, which Artikis *et al.* also use, however, the approach implements normative constraints rather than social commitments.

The normative approach distinguishes *valid* behaviour — behaviour that an agent had the power to perform at the time — from *invalid* behaviour — anything else. Interaction protocols are specified as actions, in which, for an agent Ag, and an action Act, Ag is *permitted* to perform Act, written $Permitted(Ag, Act)$,

prohibited to perform Act, written $\neg Permitted(Ag, Act)$, and *obliged* to perform Act, written $Obliged(Ag, Act)$.

Similar to the commitment machines approach, agents create and manipulate norms as a result of sending messages, thus giving meaning to the protocol using norms. The *Causal Calculator* is used in [2] to execute the specifications.

We note the following properties of this approach

- Protocols are specified from a global rather than local perspective.
- Message sequencing is specified in a declarative manner, rather than an algebraic/operational manner.
- The language used for specifying the message sequences and the meaning of messages are the same.
- There is some form of state — mainly the norms associated with actions.
- Artikis *et al.* do not discuss composition, but it seems likely that composition axioms could be defined.

5 Lightweight Coordination Calculus

The Lightweight Coordination Calculus (LCC) [13] is a process-algebra based language for first-class protocol specification. A protocol consists of a protocol definition, and a set of axioms, K, keeping track of the common information known to all participants. A protocol definition consists of a set of at least two agent clauses, $A^{\{n\}}$, with each clause defining the agents from a local participant's view. An agent clause is defined using the format $\mathbf{agent}(R, Id) ::= op$, in which R is a role name, Id is an agent identifier, and op is an operation. Operations define the protocol that an agent must adhere to, and their syntax is defined as follows:

$$
\begin{array}{lll}
op \in \text{Operation} :: \text{no op} & & \\
\quad | \ (M \Rightarrow \mathbf{agent}(R, Id)) \leftarrow \psi & & \text{(Send)} \\
\quad | \ \psi \leftarrow (M \Leftarrow \mathbf{agent}(R, Id)) & & \text{(Receive)} \\
\quad | \ op1 \ \texttt{then} \ op2 & & \text{(Sequence)} \\
\quad | \ op1 \ \texttt{or} \ op2 & & \text{(Choice)} \\
\quad | \ \mathbf{agent}(R, Id) & & \text{(Substitution)} \\
M \in \text{Message} :: \langle m, \mathcal{P} \rangle & &
\end{array}
$$

We briefly discuss this definition. 'no op' is an empty operation, meaning that the agent does nothing. The send and receive operations define the sending of a message M to the agent defined by clause $\mathbf{agent}(R, Id)$, provided that the proposition ψ is satisfiable from the common knowledge, K, and the receiving of a message M from the agent $\mathbf{agent}(R, Id)$, which results in ψ being added to the common knowledge, K. Omitting $\psi \leftarrow$ and $\leftarrow \psi$ is equivalent to specifying that ψ is true. A message is defined as a tuple $\langle m, \mathcal{P} \rangle$, in which m is the message content, and $\mathcal{P}$ is the protocol definition (written using the LCC language) that remains to be executed and the axioms of common knowledge. Composition of protocols is defined using the composition operators, `then` and `or`, which

represent sequential composition (the left operation must occur before the right), and choice (one and only one operation should occur). Finally, one can reference the name of an agent class $\mathbf{agent}(R, Id)$, and the corresponding definition (if it exists) is substituted for $\mathbf{agent}(R, Id)$. Intuitively, this represents an agent adopting the role R.

Constraints can fortify or clarify semantics of the protocols. Those occurring on the left of the '$\leftarrow$' are postconditions and those occurring on the right are preconditions. For example, an agent receiving a protocol with the constraint to believe a proposition s upon being informed of s can infer that the agent sending the protocol has a particular semantic interpretation of the act of informing other agents of propositions. This operation, $(M \Rightarrow \mathbf{agent}(R, Id)) \leftarrow \psi$, is understood to mean that message M is being sent to the agent defined as $\mathbf{agent}(R, Id)$ on the condition that ψ is satisfiable. This operation, $\psi \leftarrow (M \Leftarrow \mathbf{agent}(R, Id))$, means that once the message M is received from agent $\mathbf{agent}(R, Id)$, ψ holds. These together represent the meaning of the sending and receiving of individual messages. The meaning of a composite protocol is derived from the meaning of the messages that comprise it.

The properties of the LCC language are summarised below.

- LCC is based on the process calculus, CSP, a formal model for modelling concurrent systems. This makes LCC well suited as a language for interaction protocols and the concurrency found in multi-agent systems.
- There is long pedigree of process calculi for use as a high-level description of interactions. Besides facilitating human readability, there is a wealth of research to draw upon and apply to the field of agent coordination.
- Protocol specifications in LCC are local, rather than global.
- Although the framework provides the representation of the trace of messages occurring, there is no explicit labelling of states.
- The language for specifying message sequencing is independent of the underlying communication language.
- Constraints are declarations, not definitions.
- Designed to have a light-weight engineering requirement.
- Requires a meta-level operations to be composable.

6 $\mathcal{RASA}$

The $\mathcal{RASA}$ [11] language, part of the larger $\mathcal{RASA}$ framework, combines constraints and process algebra to model interaction protocols as first-class entities. The process algebra in the language is used to specify the sequencing of messages in a protocol, while the underlying constraint language is used to describe the meaning of messages and the message content. The meaning of entire protocols can be compositionally determined from combining the two.

Similar to LCC, $\mathcal{RASA}$'s protocol specification language resembles that of many process algebras, and in fact, the $\mathcal{RASA}$ syntax and semantics were influenced by LCC.

Let ϕ represent constraints defined in the constraint language, c communication channels, N protocol names, and x a sequence of variables. Protocol definitions adhere to the following grammar.

$$\pi ::= \phi \rightarrow \epsilon \mid \phi \xrightarrow{c(i,j).\phi} \phi \mid \pi;\pi \mid \pi \cup \pi \mid N(x) \mid \mathbf{var}_x^{\phi} \cdot \pi$$

We use π as a meta-variable to refer to protocols; subscripts and superscripts are used to denote distinct meta-variables. $\phi \rightarrow \epsilon$ represents the empty protocol, in which no message is sent and there is no change to the protocol state, but only if ϕ holds in the current state. A protocol of the format $\phi \xrightarrow{c(i,j).\phi_m} \phi'$ is an atomic protocol. It represents that i can send the constraint ϕ_m to j over channel c only if the precondition ϕ holds in the current state, in which i and j are values in the constraint language. After the message is sent, the new state of the protocol is updated using the postcondition ϕ'. This is used to specify meaning of protocols: the precondition represents a rule for a protocol because ϕ_m can only be sent if this precondition is true; and the postcondition represents the effect that sending ϕ_m has on the state.

The protocol $\pi_1;\pi_2$ denotes the sequential composition of two protocols, such that all of protocol π_1 is executed, then protocol π_2. The protocol $\pi_1 \cup \pi_2$ denotes a choice of two protocols. $N(x)$ denotes a reference to a protocol π named $N(y)$, with variables y renamed to x, such that any occurrence of N is equivalent to its definition, π. The protocol $\mathbf{var}_x^{\phi} \cdot \pi$ denotes the declaration of a local variable x, with the constraints ϕ on x. The scope of x is limited to the protocol π, and the constraints on x do not change throughout its scope; that is, x is a constant.

A *protocol specification* is defined as a set of definitions of the form:

$$N(y_1, \ldots, y_n) \mathrel{\widehat{=}} \pi$$

in which $N, y_1, \ldots, y_n$ are variable names from the underlying constraint language, and π is a protocol definition. Protocol definitions can reference other protocols in the specification using their names.

The reader may have already noted several properties of $\mathcal{RASA}$:

- The use of a process algebra inherits many of the benefits stated in Section 5.
- Protocol specifications in $\mathcal{RASA}$ are global, rather than local.
- The language for specifying message sequencing is algebraic, rather than declarative; a design decision which was made to simplify protocol composition — especially runtime composition.
- The underlying language for specifying meaning is declarative.
- The language for specifying message sequencing is independent of the underlying communication language.
- $\mathcal{RASA}$ specifications maintain a *state*, which is not explicitly sent in messages (unless this is specified as part of the messages themselves).
- The operators for protocol composition have the same syntax and semantics at all levels of dialog. That is, atomic protocols are protocols in their own right, and composing them together brings about compound protocols, which can be further composed using the same operators.

7 Petri Nets

De Silva *et al.* [3] have experimented with specification of first-class interaction protocols using Petri Nets. Petri Nets are graph structures with additional annotations. The approach proposed by De Silva *et al.* models protocols by representing the arcs of a Petri Net as possible messages, and the nodes as states between messages. Petri Nets were chosen rather than similar approaches such as finite state automata due to their ability to model concurrency.

In addition to representing messages, the approach enables the specification of *internal actions*, also using Petri Nets, which specify the actions other than message sending that agents can use. These actions include the functions an agent should execute, which variables to update after a transition, and which conditions the agent must test before sending a message. As such, these actions are used to specify the rules of the protocol, and the meanings of messages.

A local-view approach is taken in the modelling of the protocols, although it is straightforward to see how Petri Nets could also be used to specify a global view. For the local view, four types of actions (two external and two internal) are available for specifying protocols: the internal actions, *Send* and *Recv*, for sending and receiving messages respectively; and the external actions, *Action*, for reading and writing variables and executing functions, and *Pred*, which are boolean functions. These are defined as *templates*, and each must adhere to the following format:

```
Send[Sender,Performative,Receiver,Content]
Recv[Receive,Performative,Sender,Content]
Action[Label,Type,Act,Args]
Pred[Boolean]
```

The templates for *Send*, *Recv*, and *Pred* are straightforward to follow. For *Action*, *Label* is a unique label identifying the action, *Type* is *execute* for functions *read* or *write* for variables, and *Args* specifies the arguments to the function, or the values for the variables.

De Silva *et al.* do not discuss the language that is used to specify the message content, the boolean functions, or the arguments, though from examples in [3], one infers that they use some form of propositional logic. Because there appears to be no restriction on the language, it seems reasonable to say that any language capable of expressing boolean expressions could be used.

We note the following properties of this approach:

- Protocol specifications are local rather than global.
- The language for specifying message sequencing is operational, rather than declarative.
- Specifying the meaning of messages is done using a declarative language.
- Petri Nets for specifying message sequencing are independent of the underlying communication language.
- It appears that the specification must maintain state, although there is not discussion of this by De Silva *et al.* .

8 Comparisons

The approaches to first-class protocols described in the previous sections share the properties of being formally defined, meaningful, inspectable, executable and dynamically composable. However there are issues of design in which they differ. The point of this comparison is not to declare one approach as the winner but to highlight the advantages and disadvantage each. No disadvantage should be considered fatal, but merely a consideration that must be taken. It is unlikely any first-class protocol language will be the panacea to all the ills of agent communication. By highlighting the issues and differences, it is hoped that the system designer can make an informed decision when choosing to take advantage of the first-class protocol approach.

8.1 Declarative vs Algebraic/Operational

Singh [14] and Winikoff [16] both present good arguments for the benefits of declaratively specifying protocols, stating that this allows for a more flexible interaction. They argue that specifying *what* rather than *how* gives permits a more flexible approach to interaction. For example, one can specify that three events, a, b, and c, occur, and that b must occur before c, but with no other constraints. The interacting agents are free to choose the sequence of these messages as long as they obey the one constraint, which allows flexible interaction. For an algebraic language to specify this, one would likely have to specify all the possible sequences, which could lead to a larger expression. It is difficult to envisage an example that would be straightforward using a declarative language, but complex in an algebraic language, however, it is clear that a further level of abstraction provides the usual benefits associated with abstraction.

Another difference between the two approaches is the computational aspects. Adapting a declarative language would likely have the benefit that the language has tool support for automated reasoning and execution. While LCC and $\mathcal{RASA}$ are both executable if the agents can execute the underlying language, one must implement an agent to understand the process algebra in each. Furthermore, LCC and $\mathcal{RASA}$ were both designed to be quite generic, so there is no commitment to an underlying language, and tool support would be difficult to provide without committing to a particular underlying language. However, computationally, the declarative approach would be more demanding. Calculating the set of possible dialogs is straightforward in algebraic languages: simply traverse the tree that is formed by the definition. Using a declarative language, one would have to solve the paths as a constraint, which, for protocols of more than a few messages, could prove demanding. Winikoff [16] avoids this problem by implementing agents as a mapping from the protocols, however, Yolum and Singh's agents [18] reason by calculating all possible interactions.

The authors believe that a key benefit to using algebraic languages is the human readability. Despite the motivations behind first-class protocols being machine readable, it is clear that human designers will need to read and reason about these as well. An algebraic formalism is at a level that is more inline with

the way humans think about interaction. One only has to look at existing work on dialogue games [9], abstract models of interaction [7], and token-based approaches such as AgentUML to see that operational-based approaches are the favoured approach for protocol specification and design. Even outside of computer science, instructions that are meant to be read by humans, such as recipes and installation instructions, are presented in a step-by-step manner. From the literature, it seems that declarative approaches (whether first-class or otherwise) are considerably more verbose than algebraic/operational approaches. This is not surprising, because one is specifying the semantics of sequential composition, choice, etc., each time they specify such a composition. As an example of the verbosity of declarative approaches, consider the model of the Contract-Net Protocol using Social Integrity Constraints in [1]. This model consists of 17 rules, which is verbose for such a straightforward protocol, especially as the messages contain no meaning.

Finally, we note that LCC, $\mathcal{RASA}$, and the Petri Nets approach all have the advantage of not mixing the communication language with the language for message sequencing. Adapting a declarative language for modelling interaction enforces the restriction that message meaning must be specified in that language. Considering that the meaning of the message is tied in with the message itself, this further implies that all communication would also be in this language — an unfortunate restriction. This reduces the application of the language and any protocols specified in it, as demonstrated by the commitment machines approach being implemented in three different languages, the Event Calculus [18], OWL-P [4], and MAD-P [5], all by the same research group.

8.2 Local vs Global

This dichotomy is between the perspective from which the protocols are defined. *Local* protocols define clauses with respect to the dialogical activities of a single actor. For agents to communicate they must each have a set of complementary protocols –e.g. For a message being sent in one protocol, there is a message being received in another. *Global* protocols are defined as one protocol for the actions of every participant.

There are advantages and disadvantages that must be considered with respect to the perspective used by the protocol language. Local protocols have the advantage of simplicity of use for the individual agents. They do not need to sift through the protocol to determine what roles and actions apply to them. However this can obscure the activities of dialogical partners. This shortcoming can be overcome, as is done in LCC, by sending all agent clauses to an individual agent and not just the clause it is meant to execute. Conversely, global representations give a more complete representation of the conversation space, but the agent will have protocol steps that are irrelevant to it. The choice of perspective is also influenced by the model of interaction that exists within the multi-agent system. For example mediated or managed interaction would need a global protocol. Whereas, peer-to-peer interactions would be facilitated by a local representation.

Regarding the approaches in this paper, LCC and the Petri Nets approach specify protocols from a local view, while the rest specifying protocols from a global view.

8.3 Composability

Dynamic protocols are a leap that few system designers are brave enough to take. There are a number of reasons for this. Most engineers take refuge with interaction protocols for the reliability and certainty they can provide their agent interactions. They forgive the rigidity and fragility for the safety they provide. However, as the agent paradigm matures a few researchers have recognised the inevitability and benefits of protocols that can be composed automatically at run-time. There are inherit drawbacks to allowing composability such as issues of trust (e.g. who should be allowed to make changes). However, the inclusion of this functionality does not weaken any of the composable first-class protocol languages described. Indeed there are numerous frameworks where the modification of its first-class protocols is disallowed.

Although not explicitly explored, it is imaginable that a set of meta-rules could be defined over the normative approach and commitment machines to produce the composition. The same is true of the Petri Net approach. However at the current time, this has not been explored as far as the authors are aware.

LCC, although not initially designed for the purpose, has been shown suitable for dynamic composition of protocols using a number of approaches. Using dialogue games as a semantic model for composition, [10] composes the protocols for atomic dialogue games to create more complex games according to the rules of iteration, sequencing, parallelism and embedding proposed in [8]. Additionally, in [10], using adjacency pairs, composition is done at the individual message level rather than for whole protocols. The $\mathcal{RASA}$ language, in reaction to dynamic LCC, was designed with dynamic composition as a fundamental feature of the language. As a consequence, there is a much more methodical application and execution of dynamic composition in this language.

From the authors' view, we believe that algebraic languages are far more suited for composition. In algebraic languages, the start and end of a protocol and its sub-protocols are straightforward to identify. We assert that this makes compositions easier to define, and their meaning more straightforward to calculate.

8.4 Top-Down vs Bottom-Up

The top-down vs bottom-up debate is merely an issue of taste. Nonetheless, we believe that it is an important point to note, and there are good reasons why the different approaches are taken. $\mathcal{RASA}$'s bottom-up approach is a consequence of it being purposefully designed for dynamic composition, and LCC's top-down design comes from its pedigree of trying to execute electronic institutions in a more peer to peer manner. The commitment machines approaches are declaratively specified, so they adopt neither approach.

8.5 State vs No State

LCC is the only language that does not specify the meaning of messages as the alteration of a state. Instead, LCC agents explicitly pass around any such information as part of each message. The benefits of this are clear: all participants are aware of any information they need, and there is no chance that the participants view of the state can become inconsistent with each other. We see no restriction in other approaches that would prevent protocol designers from enforcing that agents send the state with each message. However, the approach is more straightforward in LCC. Two obvious disadvantages of passing the state with every message is that there is an extra overhead, and that it becomes more laborious to model protocols in which the participants should have different information.

State is important for composition. To clearly define the meaning of a compound protocol, one must relate the meaning between its sub-protocols. It seems that some form state is the only way to do. Whether this in the form of a state itself, or whether it is information that is passed, as in LCC, does not appear to be important.

8.6 Expressiveness

A comparison of expressiveness is non-trivial, because no formal framework exists for comparing the expressiveness of protocol languages. However, we note some interesting aspects of the expressiveness of first-class protocol languages.

Regarding message sequencing, the languages are quite similar. Each of them specifies a set of possible interactions, in which each interaction is a sequence of messages with preconditions and meaning. The only aspect that we see as different is regarding parallel message sending. Versions of LCC and $\mathcal{RASA}$ define parallelism as the interleaving of messages, not as messages being sent at the same time. Declarative approaches do not suffer this restriction, because messages can be declared to be sent at the same time. For example, using the Event Calculus, one can specify that messages m_1 and m_2 are sent simultaneously:

$$Happens(m_1, t) \wedge Happens(m_2, t)$$

How this is enforced in the final system, and how straightforward it would be for agents to reason about such behaviour, is an implementation detail, and is out of the scope of this paper. In the Petri Net approach, the authors of [3] state that they specifically use Petri Nets because of its ability to handle concurrent behaviour.

The second aspect of expressiveness relates to the expressiveness of messages and their meaning. The normative approach and the different commitment machine approaches can express any messages and meaning that can be expressed in the Event Calculus, OWL, or C+ respectively, and only those messages and meaning. In contrast, the $\mathcal{RASA}$, LCC, and Petri Nets approaches do not specify a particular underlying language. This provides a greater amount of flexibility in specifying meaning compared to the other approaches, because one can choose

to model protocol meaning using different underlying languages, and are flexible enough to model commitment machines and norms.

However, the fact the one can under-specify the message sequencing implies that, for flexible protocol execution, the expressiveness of declarative languages are better suited than algebraic/operational languages.

8.7 Discussion

As a reader of this paper, one may be wishing to decide which first-class protocol approach they would should use. We refrain from making any definite recommendations because the approach used is dependent on the application in which the protocols will be used, and different engineers will have different views. However, we use the results in this section to highlight three aspects of these approaches that stand out:

Local vs. Global: For a peer-to-peer interactions with two-parties, we believe a local approach is best suited. For mediated interactions, or interactions with more than two parties, we believe a global approach is best suited.

Flexible interaction: Declarative approaches seem better suited for flexible interaction; that is, under-specifying the protocol, and having agents calculate the allowable sequences.

Runtime composition: We recommend an algebraic approach if runtime composition is an important theme in an application.

9 Other Approaches

Several other approaches exist for agent interaction specification that resemble first-class protocols, but which we do not consider to be first-class approaches. In this section, we briefly introduce some of the most closely related approaches, and discuss why they do not fall into the category of first-class protocol specification languages.

Social Integrity Constraints (SICs) [1] are rules specifying actual and expected behaviour. SICs are not considered as first-class protocols because there is no *meaning* to the messages. The rules specify only which messages can occur, and the order they can occur in. In [1], the authors envisage systems in which participating agents inherently know the meaning of communicative actions, which first-class protocols aim to prevent. We believe that it would be possible for consequents of the rules to carry additional information that specified meaning, and agents could reason over this information. However, SICs do not include the notion of *state*, therefore, a new semantics would have to be specified, and would have to take into account message meaning, such as when two messages occur one after the other, but with conflicting outcomes.

Fornara and Colombetti [6] propose a method for defining the semantics of agent communication languages using commitments. Their notion of commitment is similar to that used in commitment machines, described in Section 3. Though the motivation for social commitments is similar to first-class protocols,

it is used to define the semantics of performative-based agent communication languages, rather than protocols. Composition is obtained using interaction diagrams, and the semantics of this composition is not defined formally.

10 Conclusions

The purpose of this paper was to bring more coherence to the emerging research on first-class protocols. By the comparison of the most prominent approaches, we tease out their commonalities and their differences. As this approach to agent communication inevitably attracts more interest it is conceivable that new languages will be developed, but no matter how exotic, they will have the properties described in Section 2, as well as falling on one side or the other the comparison criteria.

Though we have focused on agent communication and first-class protocols for the expression of norms of agent societies, in addition to the advantages outline in the introduction, there is another point that should be stressed. The beauty of first-class protocols is that they are generically applicable. In [12], the authors show the use of $\mathcal{RASA}$ for hybrid interactions between agents and web-service for workflow execution in the e-science domain. This is due to the power and expressiveness of first-class protocols. Social semantics are no longer held internally. They are rewritten modularly and separate from the computational entities that would make use of them. Other research has shown how web-services can follow LCC protocols for determining message passing sequences without needing to understand the social semantics of the messages being sent.

Acknowledgements

The work presented in this paper was supported by the EU projects, ARGUGRID, (ArguGRID-IST-035200), and PIPS (EC-FP6-IST-507019).

References

1. Alberti, M., Daolio, D., Torroni, P., Gavanelli, M., Lamma, E., Mello, P.: Specification and verification of agent interaction protocols in a logic-based system. In: SAC 2004: Proceedings of the 2004 ACM symposium on Applied computing, pp. 72–78. ACM Press, New York (2004)
2. Artikis, A., Sergot, M., Pitt, J.: Specifying electronic societies with the Causal Calculator. In: Giunchiglia, F., Odell, J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 1–15. Springer, Heidelberg (2003)
3. de Silva, L.P., Winikoff, M., Liu, W.: Extending agents by transmitting protocols in open systems. In: Proceedings of the Challenges in Open Agent Systems Workshop, Melbourne, Australia (2003)
4. Desai, N., Mallya, A.U., Chopra, A.K., Singh, M.P.: OWL-P: A methodology for business process modeling and enactment. In: Workshop on Agent Oriented Information Systems, pp. 50–57 (July 2005)

5. Desai, N., Singh, M.P.: A modular action description language for protocol composition. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 962–967. AAAI Press, Menlo Park (2007)
6. Fornara, N., Colombetti, M.: A commitment-based approach to agent communication. Applied Artificial Intelligence 18(9–10), 853–866 (2004)
7. Johnson, M.W., McBurney, P., Parsons, S.: A mathematical model of dialog. Electronic Notes in Theoretical Computer Science 141(5), 33–48 (2005)
8. McBurney, P., Parsons, S.: Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information 11(3), 315–334 (2002)
9. McBurney, P., van Eijk, R., Parsons, S., Amgoud, L.: A dialogue-game protocol for agent purchase negotiations. Journal of Autonomous Agents and Multi-Agent Systems 7(3), 235–273 (2002)
10. McGinnis, J.: On the mutability of protocols. Phd thesis, University of Edinburgh, Edinburgh, Scotland (2006)
11. Miller, T., McBurney, P.: Using constraints and process algebra for specification of first-class agent interaction protocols. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 245–264. Springer, Heidelberg (2007)
12. Miller, T., McBurney, P., McGinnis, J., Stathis, K.: First-class protocols for agent-based coordination of scientific instruments. In: 5th International Workshop on Agent-based Computing for Enterprise Collaboration (ACEC) Agent-Oriented Workflows and Services (to appear,2007)
13. Robertson, D.: Multi-agent coordination as distributed logic programming. In: Proceedings for International Conference on Logic Programming (2004)
14. Singh, M.P.: A social semantics for agent communication languages. In: Dignum, F., Greaves, M. (eds.) Issues in Agent Communication, pp. 31–45. Springer, Heidelberg (2000)
15. Strachey, C.: Fundamental concepts in programming languages. Higher-Order and Symbolic Computation 13(1), 11–49 (2000)
16. Winikoff, M.: Implementing commitment-based interactions. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) 6th International Joint Conference on Autonomous Agents and Multiagent Systems, IFAAMAS, p. 128 (2007)
17. Yolum, P., Singh, M.P.: Commitment machines. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 235–247. Springer, Heidelberg (2002)
18. Yolum, P., Singh, M.P.: Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and AI 42(1–3), 227–253 (2004)

Simulation of Negotiation Policies in Distributed Multiagent Resource Allocation

Hylke Buisman, Gijs Kruitbosch, Nadya Peek, and Ulle Endriss

Artificial Intelligence Programme
University of Amsterdam
{hbuisman,gkruitbo,npeek,ulle}@science.uva.nl

Abstract. In distributed approaches to multiagent resource allocation, the agents belonging to a society negotiate deals in small groups at a local level, driven only by their own rational interests. We can then observe and study the effects such negotiation has at the societal level, for instance in terms of the economic efficiency of the emerging allocations. Such effects may be studied either using theoretical tools or by means of simulation. In this paper, we present a new simulation platform that can be used to compare the effects of different negotiation policies and we report on initial experiments aimed at gaining a deeper understanding of the dynamics of distributed multiagent resource allocation.

1 Introduction

Many complex application domains can be modelled as multiagent systems in which agents of varying capabilities interact. Building such artificial societies of autonomous software agents and devising suitable interaction mechanisms presents a formidable research challenge. Within such a society, agents will have to negotiate on a number of issues, including the best possible distribution of the resources available in the system amongst the individual agents. The field of *multiagent resource allocation* [1] is concerned with the design and analysis of mechanisms for finding a suitable assignment of resources to agents, given the individual interests of the agents as well as any technical constraints imposed by the system. In *distributed* approaches to multiagent resource allocation, the computational burden of the process of allocating resources is shared by all the agents in the society. In *centralised* approaches, notably combinatorial auctions [2], on the other hand, the task of computing the optimal allocation is relegated to an external entity (e.g. an auctioneer). Here we concentrate on distributed approaches, which provide a particularly rich setting in which to study interaction in multiagent systems.

The specific resource allocation framework we adopt has previously been studied by a number of authors [3, 4, 5]. It assumes that a finite number of indivisible goods needs to be allocated to a finite number of agents. Goods cannot be shared by more than one agent, and we assume that some initial allocation is given to begin with. Each agent expresses their preferences in terms of a valuation function mapping bundles of goods to the (positive) reals, and will only accept deals

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 224–239, 2008.

(possibly involving monetary side-payments) resulting in a strict increase in utility for themselves (so-called *myopic individual rationality*). Detailed definitions will be given in Section 2. We are interested in the effects such locally conducted and individually rational deals have on the agent society as a whole. In particular, we seek to understand under what circumstances a sequence of deals will converge to an allocation that would be considered optimal in view of a particular aggregation of the individual agent preferences. Here we consider both measures for economic *efficiency*, such as Pareto optimality or the sum of individual utilities, and notions of *fairness*, such as envy-freeness or the level of utility enjoyed by a society's poorest member [1, 6, 7].

Previous work has studied such convergence properties mostly from a theoretical point of view [3, 4]. Where it is *possible* to derive general theorems on (guaranteed) convergence to a socially desirable allocation, this seems indeed the best approach. However, for many realistic scenarios some of the assumptions on which the correctness of such theorems rests simply will not hold. The amount of time available to negotiate in will be limited, and therefore perhaps not sufficient to attain an optimal state. To allow any kind of deal includes allowing very complex deals involving many agents and resources, which is computationally expensive. To be able to find convergence trends, it then becomes interesting to simulate many runs of a distributed negotiation process, under similar conditions, to see whether it may be possible to make empirically founded predictions. Previous work along these lines includes that of Andersson and Sandholm [8] and Estivie and colleagues [9, 10]. The former have studied the effects of sequencing different types of deals (such as deals involving only a single resource at a time, or deals involving the swapping of two items), while the latter have concentrated on understanding under what circumstances we can expect to see fair allocations emerge when rational agents negotiate. These works offer interesting insights into the dynamics of distributed multiagent resource allocation. However, what has been missing so far is a generic simulation platform that would allow the experimenter to vary a wide range of parameters, to run simulations for different types of agent valuations and different negotiation policies, and to evaluate outcomes with respect to a range of different efficiency and fairness criteria.

In order to fill this gap, we have developed a simulation platform called the MultiAgent Distributed Resource Allocation Simulator (MADRAS). Using this platform, a user can easily generate a scenario with given numbers of agents and resources, in which the agents have their own preferences. The agents are able to negotiate amongst themselves to establish trades using money. Using such a scenario, the user is able to run a variety of experiments to see under what circumstances the agents most beneficially manage to reallocate their resources. Finally the platform provides possibilities for visualising several experiment statistics. In this paper, we introduce the MADRAS platform and report on a set of initial experiments that we have conducted using the platform.

The remainder of this paper is organised as follows: Section 2 briefly maps out the formal resource allocation framework we use and recalls a relevant result from the literature regarding the convergence of negotiation processes to a socially

optimal allocation. Then Section 3 describes the MADRAS platform, which consists of three modules: the generation of resource allocation scenarios (in particular the generation of valuation functions); the simulation of negotiation processes conforming to a chosen negotiation policy for the agents; and an experimentation support module for evaluating and visualising the data produced during simulation. A selection of the experiments we have run using MADRAS are documented in Section 4. Section 5 concludes with a brief discussion of possible directions for future work.

2 Preliminaries

In this section, we briefly review the basic definitions of the resource allocation framework we adopt and we recall a fundamental convergence result linking the negotiation behaviour of individual agents and the emergence of optimal allocations at the societal level. Full details are available elsewhere [4].

2.1 Formal Framework

Let $\mathcal{A} = \{1, \ldots, n\}$ be a set of *agents*, and let $\mathcal{R} = \{r_1, \ldots, r_m\}$ be a set of *resources* (or goods). An *allocation* $A : \mathcal{A} \to 2^{\mathcal{R}}$ is a division of the resources in $\mathcal{R}$ amongst the agents in $\mathcal{A}$. Any allocation A has to assign each resource to *exactly* one agent. Agents may have different preferences dictating which resources they want, and how much they want them. A valuation function $v : 2^{\mathcal{R}} \to \mathbb{R}$ maps any given bundle of resources to a value in real numbers (this may be restricted to the positive reals and zero). We write $v_i(A)$ for $v_i(A(i))$, the valuation assigned by agent i to the bundle it receives in allocation A.

A *deal* $\delta = (A, A')$ is defined by the transition between two allocations (before/after). This model allows for any number of resources being reallocated amongst any number of agents within a single deal. Deals may be paired with monetary *side payments*. These are modelled using a payment function $p : \mathcal{A} \to \mathbb{R}$, satisfying $\sum_{i \in \mathcal{A}} p(i) = 0$. A positive $p(i)$ indicates that agent i has to *pay*, while a negative $p(i)$ means that i will *receive* money. The *utility* enjoyed by an agent i in a given negotiation state is computed by subtracting the sum of previous payments made by i from the valuation i assigns to the *bundle* of resources it currently holds (*quasi-linear* utility).

Whether an agent will accept a given deal (including side payments) depends entirely on whether that deal seems rational to the agent. There are a number of different rationality criteria that we could consider [7]. In this paper we shall concentrate on a myopic form of *individual rationality* [3]. A deal $\delta = (A, A')$ is called individually rational (and considered acceptable) iff it increases the utility of each of the agents involved. That is, we require $v_i(A') - v_i(A) > p(i)$ for every agent i involved in the deal (non-involved agents may receive money, but cannot be required to pay anything).

In the most general case, we assume that there are no restrictions on time or computational resources: any deal that is individually rational may eventually be identified and implemented. For more realistic scenarios, besides the restriction

imposed by the agents' rationality requirements, we may also impose structural restrictions on deals. In this paper, we are going to be interested in two such classes of deals. The class of *1-resource deals* is the class of deals involving the reallocation of a single item only (and hence only two agents). The class of *bilateral deals* is the class of deals involving only two agents (but any number of resources at a time).

2.2 Convergence

Given this framework, the question arises what kinds of allocations we can expect agents to negotiate. We are interested in assessing the quality of an allocation in terms of various criteria for economic efficiency and fairness, borrowed from the literature on social choice theory and welfare economics [6, 11]. Several examples will be given in Section 3.3. For now, let us just recall the notion of *utilitarian social welfare*. The utilitarian social welfare $sw_u(A)$ of an allocation A is given by the sum of individual agent valuations:

$$sw_u(A) = \sum_{i \in \mathcal{A}} v_i(A)$$

Observe that taking past payments into account does not affect this definition (as they always add up to zero). High social welfare implies high average utility, which justifies this as a metric for assessing the quality of an allocation.

Now, what is the connection between the *local* concept of individual rationality driving negotiation and the *global* concept of social welfare? An important result establishes that *any sequence of individually rational deals will eventually result in an allocation with maximal utilitarian social welfare* [3]. That is, no central point of control is required. We can let agents negotiate in a distributed manner following only their own selfish interests and still guarantee that the system will, at some point, reach a state that would be considered optimal from a social point of view. While this may seem surprising at first, it is actually not difficult to prove. They key insight is that, in fact, a deal turns out to be individually rational iff it increases social welfare [4]. However, we stress that the above convergence result holds only if we do not place any structural restrictions on deals. For instance, if agents will only negotiate individually rational *bilateral* deals, then the social optimum may not be reachable.

3 The MADRAS Platform

This section explains the functionality of the MADRAS simulation platform for distributed resource allocation. The platform consists of three modules:

1. The *scenario generator* is used to generate problem instances, characterised by sets of agents and resources, valuation functions for these agents, and an initial allocation of resources. Scenarios may be defined manually or generated automatically (using user-defined constraints). We have also defined an XML-based language to store and communicate scenario descriptions.

Section 3.1 discusses the most challenging task falling under this module, namely the automatic generation of valuation functions.

2. The module for *negotiation simulation* reads in a scenario description and then simulates a negotiation process. How this works exactly is determined by the chosen *negotiation policy*. Such a policy fixes choices regarding the rationality criterion used by the agents, structural restrictions imposed on deals, and the search algorithms used to identify the next deal meeting the specified requirements. This will be discussed in Section 3.2. The module can save a record of the resulting negotiation process on file.
3. The *experimentation support* module reads in one or several files documenting particular negotiation runs and can produce a wide range of experimentation statistics from this data. In particular, it can be used to plot how social welfare and similar metrics develop as negotiation progresses. Examples are given in Section 3.3.

3.1 Generating Agent Valuations

We have opted for a logic-based representation of valuation functions based on weighted propositional formulas [1, 12]. In this representation, agents may express *goals* as propositional formulas over the set of atomic propositions given by the resource names $\{r_1, \ldots, r_m\}$. For example, the goal $r_1 \wedge (r_2 \vee r_3)$ indicates that the agent in question desires to obtain r_1 and at least one of r_2 and r_3. Furthermore, agents assign numerical weights to these goals. An agent's valuation for a given bundle R is then given by the sum of the weights of the goals that are satisfied by R.[1] For example, if an agent has the weighted goals $(r_1, 3)$ and $(r_1 \wedge r_2, 1)$, then they will assign value 3 to the bundle $\{r_1\}$, value 4 to the bundle $\{r_1, r_2\}$, and value 0 to both $\{r_2\}$ and the empty bundle. This logic-based representation is not only very flexible and natural, but also fully expressive and often allows for representing interesting valuation functions in a concise manner. As far as the *automatic* generation of valuation functions is concerned, the current implementation is restricted to goals that are conjunctions of atomic propositions. This is isomorphic to the so-called *k-additive form* of representing valuation functions [1, 13].

After having specified the number of agents and resources in the scenario generation module of MADRAS, the user can initiate the automatic generation of valuations. To this end, the user may manipulate the following parameters:

- The maximum length k (number of atoms in a conjunction) for all goals in the valuation function. Either a precise value can be given, or k can be taken from a user-specified uniform or normal distribution. This parameter determines the degree of synergy between different resources.
- A function specifying the number of the goals of a given length that will actually be generated. This parameter determines the range of different bundles that an agent may wish to obtain.
- A distribution from which to pick the numerical weights for our goals.

[1] Here we interpret bundles R as models of propositional logic: an atomic proposition r is taken to be *true* in a model characterised by R iff r is an element of R.

We stress that a choice of different parameters would have been possible as well. While the present implementation gives the experimenter a good degree of control and allows for the generation of a wide range of scenarios, further research is required to establish useful guidelines for generating interesting and application-relevant sets of valuations.

Similar problems have been addressed in the context of research on combinatorial auctions, in particular for the development of the combinatorial auction test suite CATS [14]. Like for our logic-based language, bids in combinatorial auctions are symbolic expressions for encoding valuation functions. CATS can generate such bids. It is intended to model realistic bidding behaviour, for different types of real-world scenarios (such as spectrum auctions or temporal scheduling), and has been developed for testing the performance of winner determination algorithms for combinatorial auctions. Unfortunately, this data cannot (at least not immediately) be used for simulating *distributed* multiagent resource allocation. One problem is the fact that CATS does not label bids with the name of the agent bidding (the reason being that this information is not relevant from the viewpoint of testing the performance of winner determination algorithms).[2]

3.2 Simulating Negotiation Policies

We emphasise that our aim has *not* been to build negotiating agents. We are only interested in *simulating* negotiation by generating sequences of deals that would be acceptable to the agents (given their valuation functions) and to evaluate how these deals affect social welfare. An important aspect for a simulation is the *negotiation policy* used. This is determined by the following parameters:

- *Rationality criterion:* What rationality criterion do agents use to decide whether a given deal is acceptable to them? At this stage, only individual rationality has been implemented.
- *Payment functions:* Are side payments allowed? If so, and if the payments are not uniquely determined by the rationality criterion, what are the exact payments that agents have to make for a given deal? At this stage, we have implemented two simple payment functions, the *globally uniform payment function* and the *locally uniform payment function* [10].
- *Structural restrictions:* What types of deals are possible? We have implemented *1-resource deals* and (a particular form of) *bilateral deals*.
- *Search algorithms:* Given the structural and rationality-related restrictions, how do we actually find a deal to implement? This requires a search algorithm. For 1-resource deals, this is not difficult: we simply search through pairs of agents (i, j) and resources r (owned by i or j) and check whether reallocating r from from one to the other agent would be individually rational (or conformant to whichever rationality criterion we wish to apply). For bilateral deals (between two randomly chosen agents i and j), we have implemented

[2] For a discussion of exploiting CATS in the context of distributed multiagent resource allocation we refer to Estivie [9].

a search algorithm that determines an *optimal partial reallocation* (OPR) of the union of the resources currently held by i and j amongst these two agents. This will be described in more detail below.

Running a simulation for a given scenario requires choosing a negotiation policy and specifying how long the simulation should run for. This could be done by providing a time limit, an upper bound on the number of new allocations, or an upper bound on the number of *attempts* of forging a deal (and hence moving to a new allocation). In MADRAS, we have opted for the latter. While running, the system will record the sequence of allocations encountered, as well as the payments made along the way. This data can later be used to calculate social welfare and other experiment statistics.

In the remainder of this section we shall outline our approach to implementing the search algorithm for the OPR negotiation policy. After having selected a pair of agents (i, j) at random, this policy attempts to find the best possible bilateral deal between i and j. That is, it will try to find a reallocation of the items held by i and j that would maximise the sum of the valuations of i and j. To find an optimal partial reallocation, we use the A* algorithm [15]. This approach is inspired by work of Sandholm on optimal algorithms for the winner determination problem in combinatorial auctions [16].

When using A*, one must define the set of states making up the search space, the range of moves between states, the goal states, and a heuristic for moving through the state space effectively. In our case, a state is characterised by the set of resources for which we have already made a decision as to which of the two agents should receive it. Initially, all resources are unallocated. Each move assigns another resource to one of the agents, and the goal state is reached when there are no more resources to allocate.

A* refers to two functions: The function g maps each state to the value (sum of valuations of the two agents) we get for the resources already allocated in that state. The heuristic function h estimates the additional value we can still expect to generate by allocating also the remaining items. A* maintains a so-called *fringe* of states in the search space, and will always pursue the state s from the fringe which maximises $g(s) + h(s)$. By a classical result, A* will be guaranteed to find the optimal allocation provided the heuristic function h is *admissible* [15]. In our context, admissibility means that h *never underestimates* the real additional value still obtainable. For the heuristic function we are using the following formula (for a state s and agents i and j):

$$h(s) = \left(\sum_{(G,\alpha)\in\Gamma_i(s)} \alpha \right) + \left(\sum_{(G,\alpha)\in\Gamma_j(s)} \alpha \right)$$

Here $\Gamma_i(s)$ is the set of weighted goals in the representation of the valuation function of agent i that are not yet satisfied in state s, but that could still be satisfied in a follow-up state (if i were to receive all remaining resources, for instance). Formally, if $s(i)$ is the set of resources allocated to i in state s and if

$U(s)$ is the set of resources not allocated to anyone in state s, then $(G, \alpha) \in \Gamma_i(s)$ iff $s(i) \not\models G$ and $s(i) \cup U(s) \models G$. That is, for the heuristic we are computing the marginal valuation for each individual agent in the most optimistic manner and then sum these up without regard for possible conflicts. As we are restricting ourselves to positively weighted conjunctions of atomic propositions, it is not difficult to see that this constitutes an admissible heuristic for A*. While being simplistic (and certainly still subject to improvements), our heuristic already results in a very significant speed-up in comparison to a simple breadth-first search and allows us to run interesting experiments.

3.3 Evaluating and Visualising Results

The third module is a *grapher* that can be used to visualise the results obtained during simulation. Specifically, we can plot the social welfare of a sequence of allocations passed through during a simulation run. This allows the experimenter to evaluate and compare different negotiation policies in view of different desiderata. As far as the quality of allocations is concerned, MADRAS allows for plotting graphs visualising the following concepts:

- *Utilitarian social welfare:* As explained in Section 2.2, this is given by the sum of individual utilities, and is a good measure for economic efficiency.
- *Egalitarian social welfare:* This is an alternative way of defining social welfare, emphasising fairness rather than efficiency. The egalitarian social welfare of a negotiation state (possibly involving past payments) is the utility assigned to that state by the least happy agent [11].
- *Elitist social welfare:* This is defined as the utility of the happiest agent [7].
- *Envy:* Another fairness criterion is envy-freeness [17]. An agent i is said to *envy* another agent j iff agent i would prefer to own agent j's bundle of resources. Envy-free allocations are difficult to obtain through distributed negotiation, and may not even exist at all. MADRAS can plot how the *degree of envy* develops as negotiation progresses, for different interpretations of that concept. For instance, we can plot the *maximum* or the *average envy* experienced by any one agent, or we can plot the *number of envious agents* in the society.

MADRAS can also generate graphs showing the number of resources held by each agent across allocations. The closer the system gets to an optimal state, the more difficult it becomes to find a possible deal. To visualise such effects, MADRAS can plot graphs showing the number of implemented reallocations per amount of attempts at finding a deal between two randomly chosen agents.

Fig. 1 is an example for the kind of graphs generated by MADRAS. It shows how three different kinds of social welfare develop as negotiation progresses. For this particular example, we have created 50 resources and only 2 agents, and the chosen negotiation policy requires agents to negotiate individually rational 1-resource deals using the locally uniform payment function (which means that payments are arranged so as to evenly distribute the social surplus generated by a deal amongst the participating agents [10]).

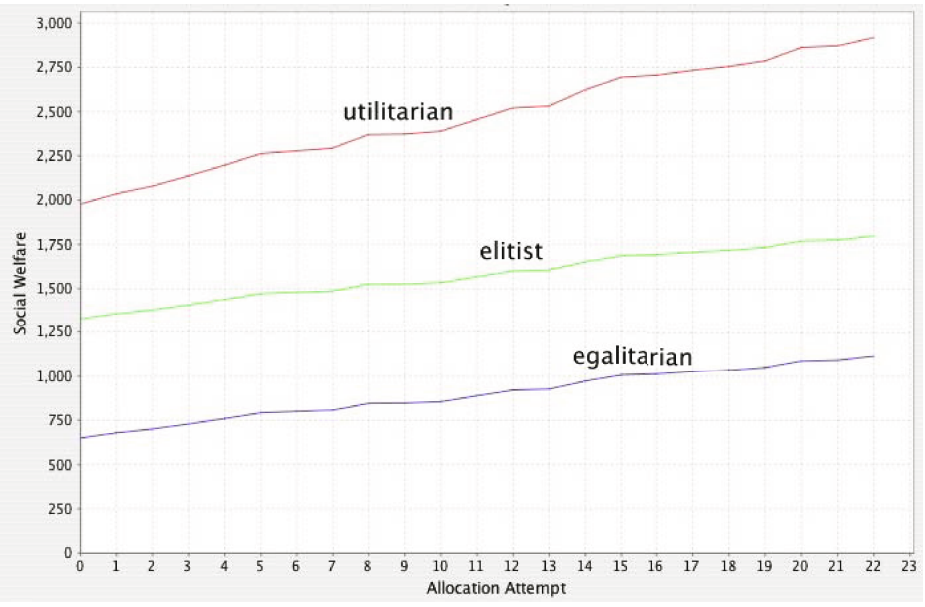


Fig. 1. Comparing utilitarian, egalitarian and elitist social welfare

Note that for the special case of a society with only two agents, the utilitarian social welfare is actually the sum of the egalitarian and the elitist social welfare, and this is clearly visible in Fig. 1. Furthermore, we can see that utilitarian social welfare monotonically increases over time, as predicted by the aforementioned result linking individual rationality and utilitarian social welfare [4]. Egalitarian and elitist social welfare are computed with respect to *utility* (rather than valuation, meaning that previous side payments are taken into account). Hence, as each deal is individually rational, also these must increase monotonically. Due to our particular choice of payment function, they furthermore increase at exactly the same rate. Hence, while egalitarian social welfare does increase, negotiation does not affect the relative fairness of the allocation: the difference in utility between the two agents does not change.

4 Experiments

In this section we report on a couple of initial experiments which we have carried out using MADRAS.

4.1 Comparing Negotiation Policies in Modular Domains

This first experiment is aimed at comparing the two negotiation policies currently implemented in MADRAS in view of reaching an allocation that maximises utilitarian social welfare when all agents are known to have *modular* valuation

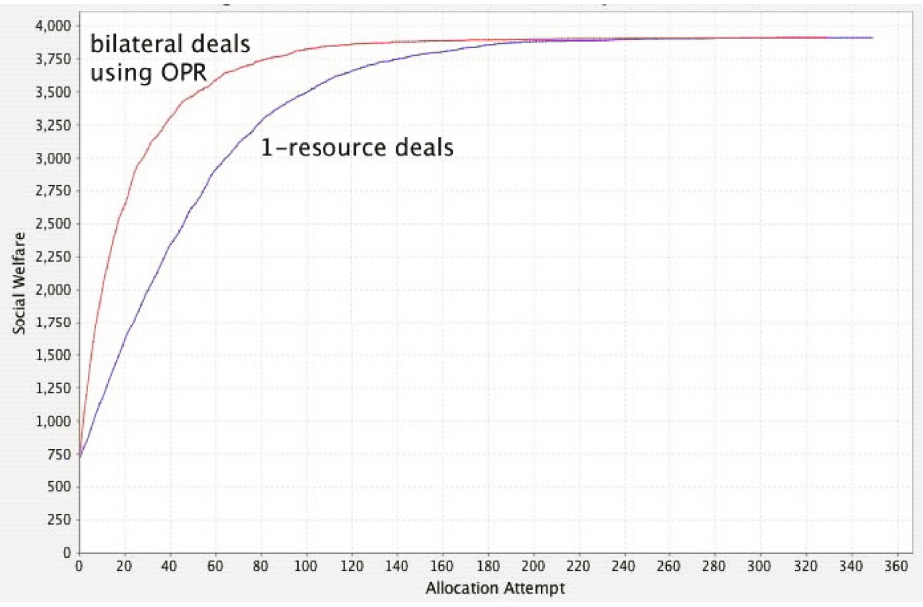


Fig. 2. Social welfare using 1-resource vs. OPR deals in modular domains

functions. Recall that a valuation function v is called modular iff it satisfies $v(R_1 \cup R_2) = v(R_1) + v(R_2) - v(R_1 \cap R_2)$ for all $R_1, R_2 \subseteq \mathcal{R}$. That is, in modular domains an agent's valuation for a given bundle R can be computed by adding up its valuations for the elements of R.

It is known that any sequence of individually rational 1-resource deals will eventually result in an allocation with maximal utilitarian social welfare, provided that all agents use modular valuation functions [4]. Given that the bilateral OPR policy subsumes the 1-resource deal policy,[3] the same must be true for the former. That is, both negotiation policies guarantee optimal outcomes in modular domains. The question is which policy does so *faster*.

Intuition suggests that the OPR policy should be faster in the sense that fewer deals are required to reach the optimum (as each individual deal can be expected to result in a greater increase in overall utility). What is not clear is how significant the difference is, and whether that advantage would not be outweighed by the fact that finding an individual deal under the OPR policy is considerably more complex than under the 1-resource deal policy (NP-complete as opposed to linear).

Fig. 2 confirms our intuitions. This experiment involves 10 agents with modular valuations over 50 resources, with each agent assigning a positive weight drawn from a uniform distribution over [1..100] to 20 randomly selected

[3] The bilateral OPR policy subsumes the 1-resource deal policy in the sense that whenever there is a 1-resource deal that would be applicable between two agents, the OPR policy will either implement that same deal or a deal that is even better.

resources. The graphs show an average of 20 experiment runs from one scenario description. Fig. 2 shows that convergence is in fact *much* faster for full bilateral negotiation using the OPR policy than for 1-resource deals, at least if "time" is measured in terms of the number of attempts made at forging an acceptable deal. Additionally, data not shown in Fig. 2 suggests that the real time required for reaching the optimum is of a similar order of magnitude for both negotiation policies. It appears that the high complexity of the search involved in computing an optimal partial reallocation in the bilateral scheme is traded off against the overhead in search required to find matching trading partners under the 1-resource policy. Of course, our findings regarding real-time performance need to be interpreted with some care: they are strongly dependent on the specific implementation choices made in the MADRAS system.

4.2 Comparing Negotiation Policies for Varying Degrees of Synergy

Our second set of experiments is aimed at comparing the performance of our two negotiation policies for varying degrees of synergy in the agent valuations. Modular valuations (as studied in Section 4.1) are representable as sets of weighted goals, each of which has length $k = 1$. If we allow proper conjunctions in the goals (of length $k > 1$), then this may be understood as synergies between the items occurring together in the same conjunction. For instance, if an agent has the goal $(r_1 \wedge r_2, 5)$, they will only receive the value of 5 if they own both of r_1 and r_2 *together*; the individual items by themselves may have no value at all.

We have produced two groups of experiments for valuation functions represented by sets of goals of length $\leq k$, with k ranging from 1 to 6. The results are shown in Figures 3 and 4, respectively. As before, there are 10 agents and 50 resources. For each value of k, we have generated 3 different scenarios and run 10 simulations for each of the two negotiation policies for each such scenario (so each of the curves shown represents the average of 30 runs). The only difference between the two groups of experiments, corresponding to Figures 3 and 4, concerns the number of weighted goals generated for each agent. In the case of Fig. 3, we have generated 20 goals of each of the required lengths for each agent. So, for instance, if $k = 3$ then an agent will have 20 goals of length 1, 20 goals of length 2, and 20 goals of length 3. All weights are drawn independently from a uniform distribution over [1..100]. In the case of Fig. 4, we have generated 30 goals *in total* for each agent (32 in the case of $k = 4$). For instance, for $k = 2$ we have generated 15 goals of length 1 and 15 goals of length 2; while for $k = 3$ we have generated only 10 goals of each length. For each pair of curves, the upper curve (better performance) corresponds to the OPR policy, and the other one to the 1-resource policy. In Fig. 4 the pairs are clearly visible as such; in Fig. 3 we have included some additional markers, which also indicate the maximum level of utilitarian social welfare achieved by each policy.

The experiments reveal some very interesting, and arguably surprising, effects. We know that for $k = 1$ (modular valuations), both negotiation policies will reach the same (optimal) state and that the OPR policy can be expected to get there in fewer steps than the 1-resource deal policy. This is visible in both

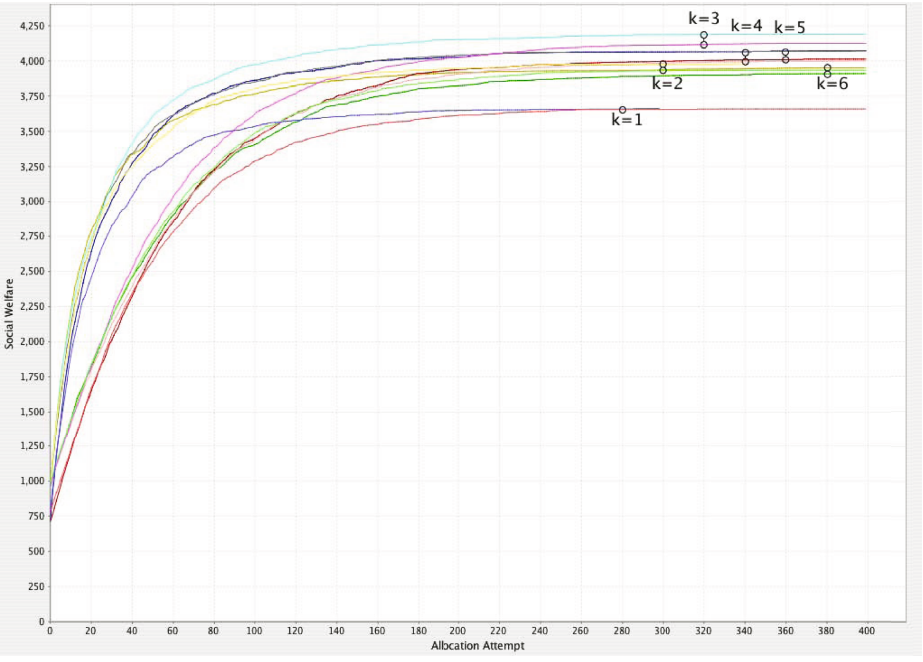


Fig. 3. Results when the number of goals per agent is proportional to k

figures. Now, as k increases (as valuations move further away from the simple modular case), we would have expected that the much more sophisticated OPR policy would outperform 1-resource negotiation even more significantly. For both policies, we would not expect to be able to reach an optimal state anymore (and this is indeed the case; data not shown here), but we would expect OPR deals to typically converge to a state with (maybe much) higher utilitarian social welfare than is attainable through 1-resource deals alone. As it turns out, this is the case only to a very limited extent. In Fig. 3, we can see that the gap between OPR and 1-resource increases as k increases up to $k = 3$, but then it becomes smaller again. So besides the expected trend described above, there must also be a second trend causing this gap to decrease again for larger values of k.

Our hypothesis is that this trend can be explained by the fact that the longer a goal, the lower the probability that all the required resources can be found in the set of items owned by the two agents supposed to forge a deal. Hence, having large amounts of long goals available in addition to the short goals present in all the scenarios actually has very little effect on the outcomes. In fact, the presence of long goals may even be detrimental to achieving high social welfare (at least if the weights for goals of any length are drawn from the same distribution, as is the case for our experiments). The reason is that satisfying a single long goal may prevent a whole set of shorter goals (of other agents) from being satisfied.

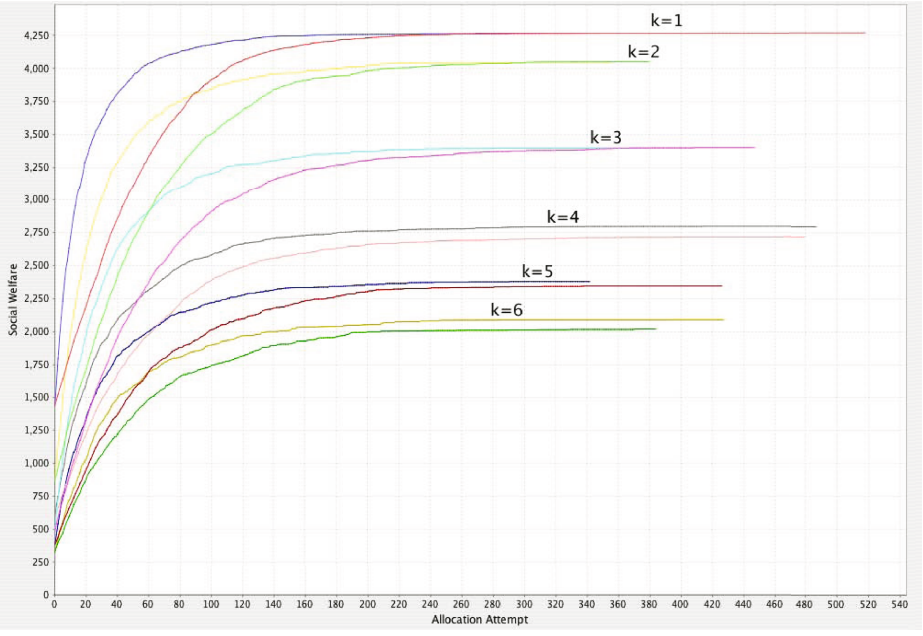


Fig. 4. Results when the total number of goals per agent is constant

In Fig. 4, the reduction in the gap between the two policies is less visible, but in any case it is still surprisingly small for larger values of k. Here we can also clearly observe a second effect: the attainable social welfare goes down as k increases. We expect this to be a consequence of there being fewer short goals in the scenarios with larger k (for Fig. 4 the *total* number of goals is constant, so the more different lengths there are, the fewer goals there are per length). These short goals are the easiest to satisfy, so the more there are the higher the sum of utilities. Indeed, further analysis of our data reveals that goals of length greater than 3 practically never get satisfied in the final allocation, and that for goals of length 3 typically no more than 1–2% get satisfied. This means that, really, what matters are the short goals of length 1 and 2.

A tentative conclusion based on these experiments would be that *any* form of bilateral negotiation (even if as seemingly sophisticated as OPR) is unlikely to be able to reach allocations that would satisfy goals that involve three or more resources. The reason for this is that chances are low that all the required resources would be present in the set of items jointly held by a particular pair of agents before negotiation between them starts. And those improvements over the *status quo* that *are* possible by means of bilateral negotiation then also seem to be achievable by means of its most basic form, namely 1-resource negotiation. Still, the OPR policy tends to achieve those moderate results in significantly fewer negotiation steps than the 1-resource policy (in terms of the number of deals attempted).

5 Conclusion

Dividing resources amongst a society of agents who have varying preferences can become a very complex task. Approaching this problem in a distributed manner and having the agents share the computational burden of the task seems promising on the one hand, but also raises serious challenges in terms of designing suitable interaction protocols. To be able to let the agents find an optimal allocation, there are many practical issues to consider. For instance, which negotiation policies are the fastest and still guarantee convergence to an optimum? How do behavioural criteria of individual agents influence the evolution of the system? A simulation platform such as MADRAS can be useful to test hypotheses about these issues. In this paper we have presented the basic functionality of MADRAS and explained the underlying principles. We have also reported on a number of experiments carried out using MADRAS. These experiments were aimed at comparing the performance of two negotiation policies in view of reaching a state with high utilitarian social welfare. In the first policy, agents negotiate individually rational deals that involve reallocating a single resource at a time. In the second policy, pairs of agents negotiate the best possible reallocation of the resources they own together amongst themselves. Despite the limited scope of these experiments, we can offer two tentative conclusions:

- Optimal partial reallocations between two agents tend to achieve the same or a higher level of social welfare than one-resource-at-a-time negotiation, and the former tend to do so in fewer steps than the latter.
- Even sophisticated forms of bilateral negotiation (such as optimal partial reallocations) are not well adapted to negotiation in domains with high degrees of synergies between large numbers of resources. In fact, in such domains the most basic form of negotiation (1-resource deals) can often achieve results very similar to those achieved by more sophisticated bilateral negotiation (although requiring a higher number of negotiation steps).

Even when studying the theoretical aspects of multiagent resource allocation closely, we are often surprised by the data that MADRAS generates. To fully understand the implications of varying any of the parameters incorporated into MADRAS we have to both analyse them theoretically and be able to explain the behaviour they generate in practice.

Our approach may be described as a middle-way between purely theoretical studies of convergence in multiagent resource allocation [3, 4, 5, 7] and work in agent-based computational economics [18, 19]. Epstein and Axtell [18], for instance, also study the emergence of various phenomena, but they do not specifically seek to understand the mathematical laws underlying such phenomena (and indeed, these may often be too complex to be easily understood or described). Here, on the contrary, we still see a mathematical explanation of emergent phenomena as an important goal, but the simulation of negotiation processes can serve as a tool for discovering the laws of distributed resource allocation mechanisms. Understanding these laws, in turn, will allow us to build better and more robust multiagent systems.

We should stress that certain design choices and features of the implementation of MADRAS are likely to have influenced (some of) our experimental results. To what extent this is the case will require further analysis. For instance, the heuristic we use for optimal partial reallocations is very important. Using the A* algorithm does not *per se* determine how to allocate goods that are not desired by either one of the agents involved in a bilateral deal. In our current implementation these uncontested resources remain with the agent they were initially allocated to, but other solutions such as random redistribution over the two agents involved are possible as well. The specific choices made during implementation in this regard may unwillingly influence not only the runtime of the algorithm but also the quality of the final allocation. Aspects such as these will require further study before we can fully bridge the gap between theoretical findings and implementation.

In addition to the above, a large number of interesting experiments remain to be done. Future work should further explore the constraints on preferences and agent rationality that are necessary to guarantee social optima. We conclude by giving three examples for specific directions of research that are being made possible by the availability of a simulation platform such as MADRAS:

- Pigou-Dalton transfers [7, 11, 20] are deals used in attempts at reducing inequality between agents.[4] However, it is known that, in the case of indivisible resources, using Pigou-Dalton transfers alone cannot guarantee convergence to an allocation with maximal egalitarian social welfare [4]. Using MADRAS would allow us to conduct research aimed at identifying constraints under which an egalitarian optimum *will* be found.
- MADRAS provides extensive possibilities for customising the agents' preferences. An interesting course of research would be to systematically examine the influence of certain classes of valuation functions on the reachability of certain social optima. For instance, while theoretical research has provided a good understanding of convergence behaviour in either the fully general case or the very simple case of modular valuations [4], little is known about convergence by means of structurally simple deals in case of valuations that are subjected to severe restrictions other than modularity.
- MADRAS also provides for another research approach which would not be possible without such a platform. This approach is to run many "arbitrary" experiments and examine these to form hypotheses (possibly using machine learning techniques). An approach of this type may produce findings which are not intuitive and would otherwise not be encountered easily.

Acknowledgements

This work has been carried out in the context of the BSc Artificial Intelligence Honours Programme at the University of Amsterdam. MADRAS is available at `http://madras.infosyncratic.nl`.

[4] A Pigou-Dalton transfer is a deal between two agents that results in a transfer of utility from the stronger to the weaker agent, without reducing their sum of utilities.

References

1. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J., Phelps, S., Rodríguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Informatica 30, 3–31 (2006)
2. Cramton, P., Shoham, Y., Steinberg, R. (eds.): Combinatorial Auctions. MIT Press, Cambridge (2006)
3. Sandholm, T.W.: Contract types for satisficing task allocation: I Theoretical results. In: Proc. AAAI Spring Symposium: Satisficing Models (1998)
4. Endriss, U., Maudet, N., Sadri, F., Toni, F.: Negotiating socially optimal allocations of resources. Journal of Artificial Intelligence Research 25, 315–348 (2006)
5. Dunne, P.E., Wooldridge, M., Laurence, M.: The complexity of contract negotiation. Artificial Intelligence 164, 23–46 (2005)
6. Arrow, K.J., Sen, A.K., Suzumura, K. (eds.): Handbook of Social Choice and Welfare. North-Holland, Amsterdam (2002)
7. Endriss, U., Maudet, N.: Welfare engineering in multiagent systems. In: Engineering Societies in the Agents World IV. LNCS (LNAI), vol. 3071. Springer, Heidelberg (2004)
8. Andersson, M., Sandholm, T.W.: Contract type sequencing for reallocative negotiation. In: Proc. 20th International Conference on Distributed Computing Systems (ICDCS 2000). IEEE Press, Los Alamitos (2000)
9. Estivie, S.: Allocation de Ressources Multi-Agent: Théorie et Pratique. PhD thesis, Université Paris-Dauphine (2006)
10. Estivie, S., Chevaleyre, Y., Endriss, U., Maudet, N.: How equitable is rational negotiation? In: Proc. 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006). ACM Press, New York (2006)
11. Moulin, H.: Axioms of Cooperative Decision Making. Cambridge University Press, Cambridge (1988)
12. Lang, J.: Logical preference representation and combinatorial vote. Annals of Mathematics and Artificial Intelligence 42(1–3), 37–71 (2004)
13. Grabisch, M.: k-order additive discrete fuzzy measures and their representation. Fuzzy Sets and Systems 92, 167–189 (1997)
14. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a universal test suite for combinatorial auction algorithms. In: Proc. 2nd ACM Conference on Electronic Commerce. ACM Press, New York (2000)
15. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
16. Sandholm, T.W.: Optimal winner determination algorithms. In: Cramton, P., Shoham, Y., Steinberg, R. (eds.) Combinatorial Auctions. MIT Press, Cambridge (2006)
17. Brams, S.J., Taylor, A.D.: Fair Division: From Cake-cutting to Dispute Resolution. Cambridge University Press, Cambridge (1996)
18. Epstein, J.M., Axtell, R.L.: Growing Artificial Societies: Social Science from the Bottom Up. MIT Press, Cambridge (1996)
19. Tesfatsion, L., Judd, K. (eds.): Handbook of Computational Economics: Agent-Based Computational Economics. Elsevier, Amsterdam (2006)
20. Dunne, P.E.: Extremal behaviour in multiagent contract negotiation. Journal of Artificial Intelligence Research 23, 41–78 (2005)

Collective-Based Multiagent Coordination: A Case Study

Matteo Vasirani and Sascha Ossowski

University Rey Juan Carlos, Madrid, Spain
{matteo.vasirani,sascha.ossowski}@urjc.es

Abstract. In this paper we evaluate Probability Collectives (PC) as a framework for the coordination of collectives of agents. PC allows for efficient multiagent coordination without the need of explicit acquaintance models. We selected Distributed Constraint Satisfaction as case study to evaluate the PC approach for the well-known 8-Queens problem. Two different architectural structures have been implemented, one centralized and one decentralized. We have also compared between the decentralized version of PC and ADOPT, the state of the art in distributed constraint satisfaction algorithms.

1 Introduction

In a multi-agent system (MAS) the term coordination refers to the process in which agents reason about their local actions and the (anticipated) actions of other agents in order to ensure that the community acts in a coherent manner [7], i.e. satisfy a global requirement/objective function.

While in micro-coordination, the objective of the designer is to build an agent that is able to coordinate with the existing ones, in order to satisfy its own utility, in macro-coordination the designer of the MAS has a systemic vision, and the main interest is some sort of global utility maximization.

In the case that there is a unique designer of the system, who has complete control over the agents' internal structure, the agents are said to be cooperative, and the MAS is a sort of problem solving system. Stigmergic coordination [9] and collective intelligence [12] fall in this category.

Even in the case of cooperative MASs (e.g. a swarm of explorer robots), the task of building a coherent, global utility maximizing system, starting from individual autonomous agents, is not so trivial. The fact that the designer has the control over each agent utility function and action space, is not enough to assure that the resulting system is optimal from a global perspective. It is well known that if the agents try to maximize their utilities in a greedy way (i.e. without considering the externalities of its actions), this can lead to very poor performances of the whole system (the so-called Tragedy of the Commons [6]).

The aim is so providing agents with well-designed private utilities, so that the selfish optimization of each agent utility leads to increased performance of the global utility of the collective as a whole. Thus, it is a matter of reverse

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 240–253, 2008.

engineering: the objective is to define suitable private utility functions for the individual agents, so that they "coordinate and cooperate unintentionally" and optimize the world utility.

The aim of this paper is to evaluate the characteristics of COllective INtelligence (COIN) [18,19,20] and its variation Probability Collectives (PC) [21], as coordination framework for cooperative multiagent systems. The paper is structured as follows: in section 2 we introduce the theory of Probability Collectives; in section 3 we take Distributed Constraint Satisfaction Problems (DisCSP) [15] as an example of distributed cooperative coordination; in section 4 we evaluate a centralized version of the collective to solve the 8-Queens DisCSP, while in section 5 we evaluate a decentralized version of the collective; finally in section 6 some conclusions and future works ideas are outlined.

2 Background: Probability Collectives

The problem of coordination in systems of autonomous (self-interested) agents can be easily modelled in terms of utility functions. Each agent is given a private utility function, which maps each action that the agent can take into the real numbers. The aim of the agent is to locally optimize this function. Furthermore there is a notion of global utility that represents the effectiveness of the system as a whole, regarding its design goals. The aim is to provide agents with well-designed private utilities, so that the selfish optimization of each agent's utility leads to increased performance of the global utility of the collective as a whole.

Recently, there have been studies aimed at developing guidelines regarding the properties that such private utilities functions need to meet. The theory of COllective INtelligence (COIN), developed by Wolpert et al. [18,19,20]), for instance, requires local utility functions to be *aligned* with the global utility G (i.e. if the private utility grows, the global utility does the same), as well as easily *learnable* (i.e. they enable the agent to distinguish its contribution to the global utility).

Once the collective designer has defined the private utilities, many methods are available for supporting the agents' decision making and optimization of their private utility functions. These may include evolutionary computation [1] or multiagent reinforcement learning [17]. Still, all these methods are based on some kind of search for the best action within the space of local actions. In this paper we will draw upon a novel method called Probability Collectives (PC) [21], which has been developed within the COIN framework, in which the optimization across possible *actions* is replaced with the optimization across the *space of probability distributions of possible actions*. In this way, PC allow for the distributed optimization of objective functions that are not smooth and variables that are discrete, continuous or mixed.

Generally speaking, a collective can be formalized as a set of n agents. In the simplest version, each agent i controls one system variable, x_i, which can take on finite number of values from the set X_i. So these $|X_i|$ possible values constitute

the possible moves of the i'th agent. The variable of the joint set of n agents describing the system is $x = \{x_1, x_2, \ldots, x_n\} \in X$, with $X = X_1 \times X_2 \times ... \times X_n$. Each agent is given a private utility function, $g_i(x)$, which it aims at optimizing. We remark that such utility is a function of x, the joint action of the collective, that is the utility of agent i depends also on the actions of all the agents other than i ($x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$). The utility of a joint action x at the system level is modelled by a global utility function G over X, which the collective designer aims at optimizing.

Unlike many optimization methods, in PC the underlying x is not manipulated directly, rather a probability distribution over that variable is manipulated. The ultimate goal of this approach is to induce a product distribution $q = \prod_i q_i(x_i)$ that is highly peaked around the x optimizing the objective function of the problem, and then obtaining the optimized solution x by sampling q.

When X is finite, q is a vector in an Euclidean space with real-valued components, so finding the q optimizing the expectation value $E_q(G)$ means optimizing a real-valued function of a real-valued vector. Furthermore, the PC formalism can be used for essentially any X, be it continuous, discrete or mixed [2].

2.1 Maxent Lagrangian

Since the agents do not have any explicit symbolic model of its acquaintances in the collective, the available prior knowledge is limited to their private utility functions and their expected utility values. The main result of PC is that the best estimation of the distribution q_i that generates those expected utility values is the distribution with maximum entropy. Formally, the minimum value of the global utility function[1] can be found by considering the maxent Lagrangian equation for each agent:

$$\begin{aligned} \mathcal{L}_i(q_i) &= E_q[g_i(x_i, x_{-i})] - T \cdot S(q_i) = \\ &= \textstyle\sum_{x_i} E_{q_{-i}}[g_i(x_i, x_{-i})] \cdot q_i - T \cdot S(q_i) \end{aligned} \tag{1}$$

where q_i is the probability distribution over agent i's actions x_i; T is an inverse Lagrangian multiplier, or a "temperature" that defines the balance between "exploitation" and "exploration"; $E_q[g_i(x_i, x_{-i})]$ is the expected value of the utility function, which is subjected also to all the other agents' actions, x_{-i}; and $S(q_i)$ is the Shannon entropy associated with the distribution q_i, $S(q_i) = -\sum_{x_i} q_i(x_i) \ln[q_i(x_i)]$.

2.2 Minimizing the Maxent Lagrangian

Since the maxent Lagrangian is a real-valued function, it is possible to use search methods for finding function extrema, such as gradient descent or Newton methods. These techniques minimize a function by making small steps in the direction

[1] Without loss of generality, the global utility function G is considered as a "cost" to be minimized.

of function derivative (i.e. gradient). Nearest Newton update has been proved to be one of the most effective descent rule. Each agent's probability distribution at time $t+1$ is obtained by:

$$q_i^{t+1} = q_i^t - \alpha q_i^t \times \{\frac{E_q[g_i|x_i] - E_q[g_i]}{T} + S(q_i^t) + \ln[q_i^t]\} \quad (2)$$

where $E_q[g_i]$ is the expected utility, $E_q[g_i|x_i]$ is the expected utility associated with each of the agent i's possible actions, and α is the update step.

Equation 2 shows how the agents should modify their distributions in order to jointly implement a step in the steepest descent of the Maxent Lagrangian. At any time step t, each agent i knows q_i^t exactly, and therefore knows $\ln[q_i^t]$, but it might not know the other agents' distributions. In such cases, it is not able to evaluate any expected value of g_i, since it depends on the probability distributions of all the agents.

One way to circumvent this problem is to have those expectation values be simultaneously estimated by repeated Monte Carlo sampling of the distribution q to produce a set of $(x; g_i(x))$ pairs. These pairs can then be used by each agent i to estimate the values $E_q[g_i|x_i]$, for example by uniform averaging of the g_i values of the samples associated with each x_i.

The basic algorithmic framework is explained in algorithm 1 [8]. The starting temperature depends on the problem, while the initial q is the maximum entropy distribution, i.e. the uniform distribution over the action space X. The temperature is lowered accordingly to a schedule determined by the function *updateT* (e.g. every C iterations). The minimization of $\mathcal{L}$ for a fixed temperature is accomplished by repeatedly determining all the conditional expected utility values $E_q[g_i|x_i]$ (function *evalConditionalExpectations*) and then using these in the Nearest Newton update rule (function *updateQ*). Also the convergence criteria depends on the problem, but in general the algorithm stops if the change in the probability distribution q falls below a threshold.

Algorithm 1. PC framework

```
01:  T <- initializeT
02:  q <- initializeQ
03:
04:  while not converged
05:    m  <- MCsample
06:    ce <- evalConditionalExpectations(m)
07:    q  <- updateQ(ce)
08:    T  <- updateT
09:  end while
10:
11:  return mostProbableJointMove
```

3 The Problem: DisCSP as Cooperative Coordination

In this section we relate the problem addressed by PC to Distributed Constraint Satisfaction Problems (DisCSP) [15]. In DisCSPs, a set of variables is distributed among agents, each of which can take a value from a specific domain. The variables are connected by constraints which constitute predicates over certain variables, defining the set of admissible assignments of values to variables. The search algorithm for solving these problems is a distributed algorithm, run by agents that communicate by sending and receiving messages. In general, messages contain information about assignments of values to variables and refutations of assignments.

We chose the 8-Queens problem as example of DisCSP. The problem can be formalized as follows:

- Let V be the set of variables, $V = \{v_1, \ldots, v_8\}$, each of them corresponding to a row in the chessboard.
- Each variable v_i can take a value from the domain $D_i = \{1, \ldots, 8\}$, where each value corresponds to a column of the chessboard where it is possible to place a queen.
- The constraints among variables are
  - $v_i \neq v_j$
  - $v_i - v_j \neq i - j$
  - $v_i - v_j \neq j - i$

The mapping of this problem into a collective is straightforward. The collective is composed of 8 agents, $x_1, \ldots, x_8$. Each agent x_i controls the corresponding row of the chessboard, by putting a queen in one of the 8 columns. So the moves space of each agent is $X_i = \{1, \ldots, 8\}$. Since the system goal is to minimize the number of violated constraints[2], each agent's utility function must reward the agent moves that imply few (or no) attacked queens. We remark that we use the term "utility" although it is actually a "cost" to be minimized, so along this paper low values of g correspond to good utility values.

Each agent's private utility is the so called *Wonderful Life Utility* (WLU). The value of the WLU for the agent x_i is defined as

$$g_i(x) = WLU_i(x) = G(x) - G(CL_i(x)) \tag{3}$$

where x if the joint action of the collective and $CL_i(x)$ is the "virtual" joint action formed by replacing the i-component of x to an arbitrary fixed value, e.g $\overrightarrow{0}$. In this way WLU is equivalent to the world utility minus the world utility that would have arisen if agent x_i "had never existed".

Such an utility has been demonstrated [18] to be both highly learnable and aligned with the global utility. Since the second term in equation 3 does not depend on the action that takes agent i, any action that improves $WLU_i(x)$ also improves the global utility $G(x)$.

[2] We remark that we allow also non-complete (suboptimal) solution to the problem, differently from traditional constraint satisfaction.

In the case of the 8-Queens problem, the WLU for agent x_i is defined as follows:

$$g_i(x) = WLU_i(x) = G(x) - G(CL_i(x)) = \mu \cdot e^{\mu} - \widehat{\mu} \cdot e^{\widehat{\mu}} \quad (4)$$

where μ is the number of attacked queens in the chessboard configuration that results from the joint action of the collective, and $\widehat{\mu}$ is the number of attacked queens if x_i "disappears" from the collective, i.e. it doesn't put its queen on the chessboard. The best value for a so designed utility is 0, i.e. the number of attacked queens doesn't change if agent x_i disappears from the collective. This means that agent x_i did the best possible move, that is placing its queen without attacking any other queen.

4 Centralized Implementation

In this section we show how the basic algorithmic framework (see algorithm 1) can be applied to our scenario. As said in section 2, in order to implement a collective, several design parameters need to be fixed, like the initial temperature T, the annealing policy, the update step α of the Nearest Newton scheme. Furthermore, a key point in the implementation is the architectural structure of the agents, i.e. the structure that enables the agent to construct the set of sampled joint moves. In literature, the typical implementation is letting all the agents at time t contribute to the creation of the sampled joint moves set, by sampling its own distribution, and then using the sampled joint moves set to update their distribution.

This architectural structure doesn't require explicit synchronization, in the sense that no agents can be idle, or waiting for synchronization messages from other agents. However, even if all the agents are simultaneously active, it is necessary that they share the same "clock". So the collective as a whole can be seen as a two state automaton, switching between a sampling phase and a distribution update phase, and all the agents are simultaneously in the same phase.

For our purposes, we deployed a system (see figure 1), based on Jade[3], composed of the collective agents and a shared environment. The agents can access the environment both in "write" and "read" mode, that is they write when they put their samples in the shared environment, while they read when they evaluate the joint moves.

In the initialization phase, all the agents set the temperature T to the same value, and they initialize their own distributions q_i to the uniform one.

At each iteration, the agents put their samples in the shared environment and use the joint moves to update their distribution, following the Nearest Newton update rule (see section 2.2).

The annealing policy was multiplying the temperature by an annealing rate between 0 and 1 after a fixed number of iterations, where values close to 0 determine a higher annealing rate.

[3] http://jade.tilab.com

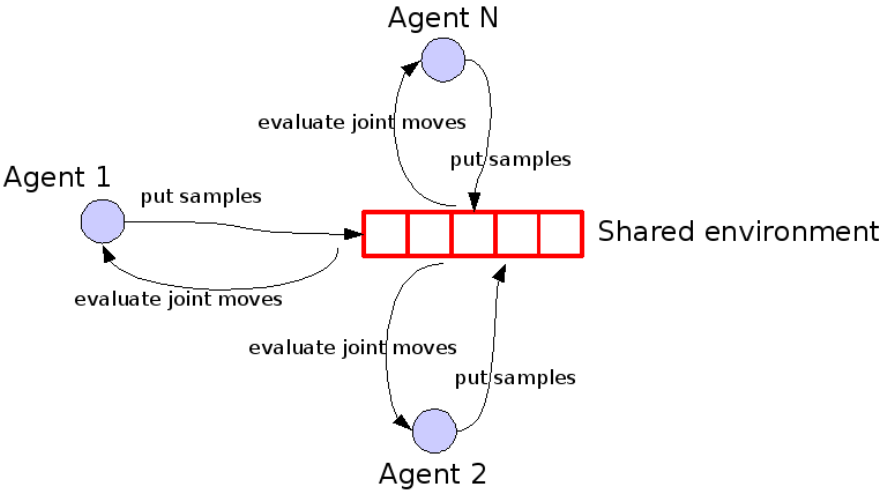


Fig. 1. Centralized implementation

The algorithm stops if one of these four conditions are met:

- The actual most probable joint move corresponds to a configuration with no attacked queens
- One of the sampled joint move corresponds to a configuration with no attacked queens
- The average mean squared difference between the old agents' probability distributions and the updated ones falls below a threshold
- The average temperature of the agents falls below a threshold.

4.1 Experimental Results

As said in section 2, the implementation of the collective depends on several design parameters, like the initial temperature T, the annealing rate, and the update step α of the Nearest Newton update rule.

The aim of the experiments was finding the parameterization that gives the best results, in terms of iterations to the convergence and quality of the solutions found by the collective, expressed by the number of attacked queens.

Along all the experiments, we kept fixed the convergence criterion and the initial temperature, while we made tests with different combinations of the annealing rate and the update step α.

The first experimental setup (Test 1) was characterized by an annealing rate of 0.5 and an update step of 0.2. The average number of iterations to the convergence was 78.3 and the average global utility (expressed as the number of attacked queens) was 0.6.

The second experimental setup (Test 2) was characterized by an annealing rate of 0.1 and an update step of 0.8. The average number of iterations to the

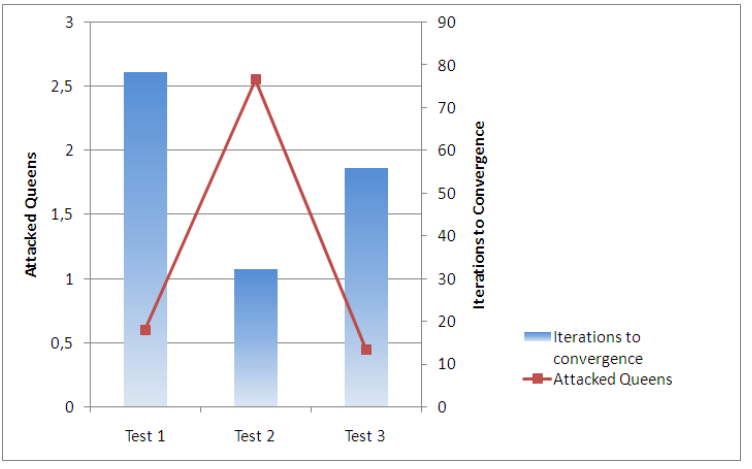


Fig. 2. Average number of iterations to convergence and average number of attacked queens for the 3 configurations

convergence was much lower, 32.15, but the quality of the solutions found was worse, since the the average average global utility was 2.55.

This kind of result was expected because this parameters setup is equivalent to a "higher" speed in the update rule. For example, with a lower annealing rate, the temperature decreases more rapidly. As seen in equation 2, high temperatures induces small changes in the probability distributions, guaranteeing "exploration" of the moves space and avoidance of local minima, but also slower convergence.

Similar considerations can be done for the update step α. With low values of this parameter, only a small part of the distribution update is added (or subtracted) to the old distribution, while a higher value of α provokes greater modifications to the probability distribution.

Given the effect of the combination of annealing rate and update step, we tried a third experimental setup (Test 3), with an annealing rate of 0.5 and an update step of 0.2 for all the agents except two, which have been parameterized with an annealing rate of 0.1 and an update step of 0.8. With this configuration, the average number of iterations to the convergence was lower than the first configuration, with a value of 55.85, while the quality of the solutions found was better, since the the average global utility was 0.45. Figure 2 summarizes the results.

The introduction of a certain degree of "heterogenicity" in the collective has resulted in an improvement of both the speed and the quality of the algorithm. This can be explained by the fact that the simultaneous update of the distributions may confound each other, especially if this update is only a smooth change to each agent's probability distribution.

Conversely, the fact that two agents descend the function surface more rapidly along two directions, benefited the whole collective. With the third configuration, the collective is able to find a solution for the 8-Queens problem in the 80% of the cases, with an average of 55.85 iterations per agent.

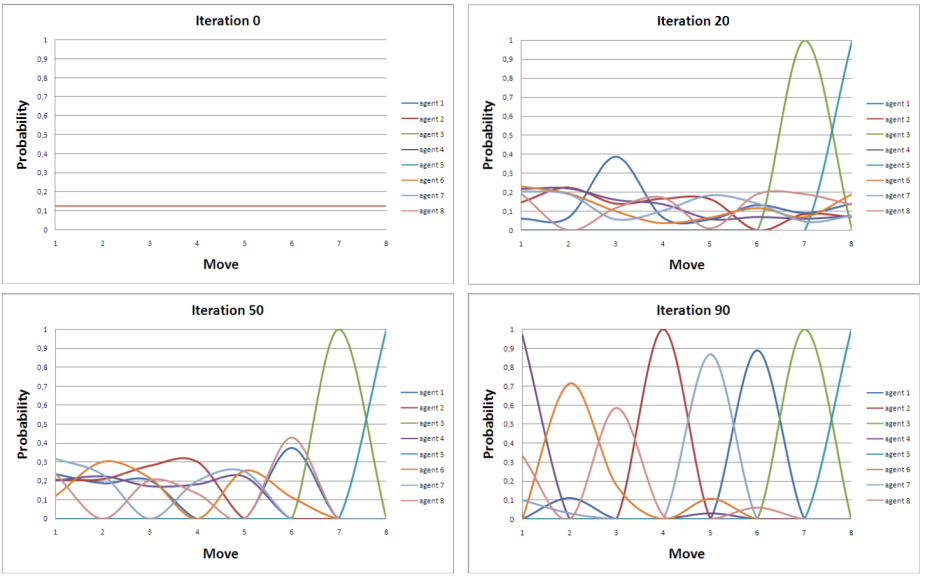


Fig. 3. How the agents' distributions change

In figure 3 we plotted the probability distributions of the 8 agents at different stages of an experiment. At the beginning, each agent has a uniform distribution over the possible moves, and then these distributions change during the execution of the experiment. The objective of the collective is inducing a probability distribution highly peaked in correspondence of the best moves, that is the moves that generates the best chessboard configuration. It is also possible to notice how the distributions of agent 3 and 5 change more rapidly then the other distributions, due to the aforementioned higher "speed" in the distribution update.

5 Decentralized Implementation

The aforementioned implementation requires a central entity to gather and successively evaluate the sampled joint moves. Furthermore, all the agents need to share the same "clock" in order to jointly create a set of sampled moves to update their distribution. However, alternative logical structures of the agents can be taken in consideration. For example in [16], the set of sampled joint moves is built using a token-ring message passing architecture.

In the second architectural structure that we implemented (see figure 4), each agent runs its program without any common "clock", and asynchronously requests to all the other agents to sample their distributions in order to create a set of joint moves and use them to update its distribution. This approach enables the samples set to be generated in a distributed manner.

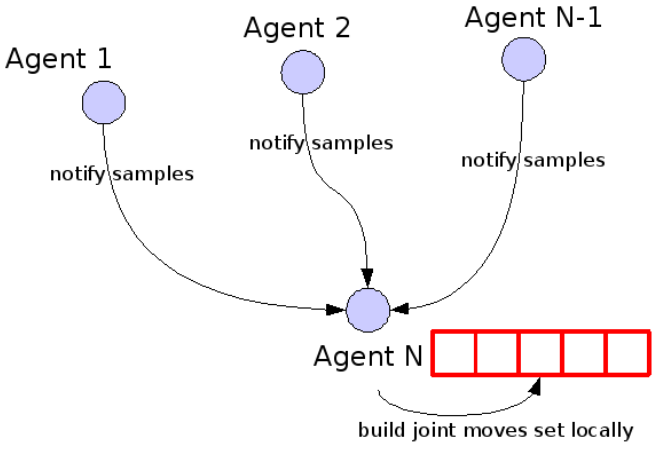


Fig. 4. Decentralized implementation

Table 1. Centralized vs decentralized implementation

	PC centralized	PC decentralized
Iterations per agent	55.85	44.52
Attacked queens	0.45	0.34

Due to the asynchronous nature of the architectural structure, it is possible that some disalignments between the agent states occur (e.g. the convergence criteria can be met by agent i but not by agent j).

Table 1 summarizes the results obtained with the centralized and decentralized implementations. We can see how the decentralized version produces on average better solutions than the centralized one. Also the number of iterations executed on average by each agent is lower than in the centralized version.

This can be explained by the fact that in the centralized version, all the agents update their distribution at time t on the basis of the probability distributions of the other agents at time $t-1$. So at time t, the information that an agent uses to take its decision is going to go out-of-date.

On the other hand, in the decentralized version, since the agents run their program asynchronously, it is reasonable to think that when an agent takes a decision at time t, the information that it uses is still valid (see figure 5).

To better evaluate the decentralized implementation, we have compared it with ADOPT [10], a distributed algorithm explicitly designed for distributed constraint satisfaction problems. We compared the quality of the solutions that the two algorithms are able to find, and the number of messages exchanged by the agents.

ADOPT is a quality guaranteeing algorithm, i.e. it always finds a solution to the problem, and needs the agents exchange the same number of messages, which in the case of the 8-Queens problem is 2792. This value depends on the

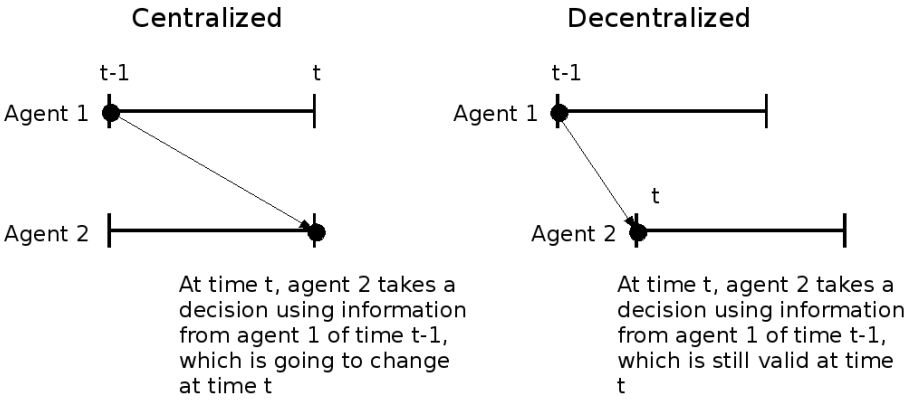


Fig. 5. Centralized vs decentralized implementation

Table 2. Comparison between ADOPT and collective coordination

	ADOPT	PC average	PC best	PC worst
Exchanged messages	2792	2676.4	176	6864
Attacked queens	0	0.34	0	2

initial configuration of the chessboard, which has been fixed to the configuration with all the queens in the first column[4].

Due to the probabilistic nature of the collective-based approach, we reported the average, best and worst case. Results are summarized in table 2.

We can see how ADOPT needs to exchange slightly more messages in order to reach a minimum, although it guarantees to reach a global one. On average the collective exchanges a similar number of messages, and is able to find a complete solution in the 86.7% of the cases. In the best case it finds a global minimum by exchanging only the 6.3% of the messages needed by ADOPT, while in the worst case it stucks in a local minimum.

In figure 6 we plotted the number of attacked queens as a function of the messages exchanged by the agents. As expected, it is possible to appreciate how the collective progressively optimizes the objective function during the execution of the algorithm. On the other hand, ADOPT shows a trend typical of a backtracking algorithm; every time a local minimum is reached, the agents change their configuration, increasing the number of attacked queens, up to find a solution of the problem.

A strength of PC is that the communication and computational load is equally distributed among all the agents, while with ADOPT, few agents send the majority of the messages. The number of messages exchanged by each agent on

[4] This initial configuration is somehow equivalent to assign uniform probability distributions to the collective agents (and not starting with biased, i.e. "peaked", distributions).

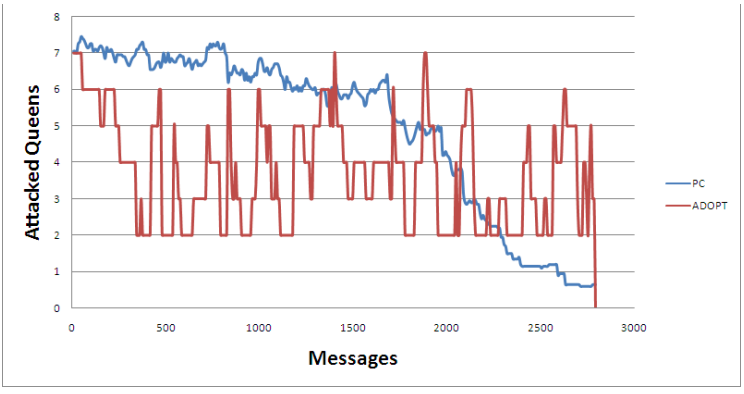


Fig. 6. How the number attacked queens decreases with the exchange of messages

average is 334.55, while the highest number of messages exchanged by a single agent is 387.2, that is only 15.74% above the average.

Furthermore the collective doesn't rely on any rigid, preprocessed structuring of the agents, and doesn't have a single point of failure. For example, even a failing agent can be easily considered as an agent that simply doesn't update its probability distribution. All these characteristics make PC robust and potentially suitable for noisy, real-world problems.

6 Conclusions

In this paper we evaluated Probability Collectives (PC) as a promising framework for the coordination of collectives of agents. We conducted experiments with the 8-Queens problem, to test the effectiveness of the coordination framework. In particular we implemented two different architectural structures, one centralized and one decentralized, and we made experiments to measure the iterations to convergence and the solutions quality.

We also make comparisons between the decentralized version and ADOPT, the state of the art in distributed constraint satisfaction algorithms, and we saw how the collective-based approach, with a similar number of exchanged messages respect to ADOPT, finds a complete solution in the 86.7% of the cases.

PC is a very general framework for agent coordination and distributed optimization, and it is not limited to DisCSP problems. This makes it not at the same level of distributed algorithms especially tailored for DisCSP, like the aforementioned ADOPT and Asynchronous Weak-commitment Search [14]. A strength of PC is that it can address a broader class of problems, from distributed optimization to distributed control [3].

PC can also be related to distributed problem solving system, like TÆMS [4]. A distributed problem solving system is a distributed network of semi-autonomous processing nodes that work together to solve a single problem. The advantage of

PC respect to such systems is that the agents are not structured accordingly to an *a priori* task decomposition, neither they need to be provided with an explicit model of the other agents goals, plans or tasks.

PC has already been applied to several real-world problems, like distributed control [3], distributed data fusion in sensor networks [16] and clustering in ad-hoc networks [13]. Our future works will be applying the coordination framework to a challenging, large-scale problem, like traffic management systems [11], since it is reasonable to expect great benefits over existing approaches, especially if the problem has a dynamic nature and involves noise and uncertainty. We will extend the framework to support multiple actions for each agent, in order to make PC widely applicable. We will also explore different techniques to estimate expected utility values, not only from statistics but also from the multi-agent world.

Acknowledgments

We would like to thank David Wolpert and Dev Rajnarayan for their help and the profitable discussions relating to Probability Collectives algorithms.

This research was partially supported by the Spanish Ministry of Science and Education through projects "THOMAS" (TIN2006-14630-C03-02) and "AT" (CONSOLIDER CSD2007-0022, INGENIO 2010).

References

1. Back, T., Schwefel, H.: Evolutionary Computation: An Overview. In: International Conference on Evolutionary Computation, pp. 20–29 (1996)
2. Bieniawski, S., Kroo, I., Wolpert, D.: Discrete, continuous, and constrained optimization using collectives. In: Proc. 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York (2004)
3. Bieniawski, S.: Distributed Optimization and Flight Control Using Collectives, PhD Thesis, Stanford University (September 2005)
4. Decker, K.: TÆMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms. In: Foundations of Distributed Artificial Intelligence, ch. 16, pp. 429–448 (1996)
5. Groves, T.: Incentives in teams. Econometrica 41, 617–631 (1973)
6. Hardin, G.: The Tragedy of the Commons. Science 162, 1243–1248 (1968)
7. Jennings, N.R.: Coordination Techniques for Distributed Artificial Intelligence. In: O'Hare, G.M.P., Jennings, N.R. (eds.) Foundations of Distributed Artificial. John Wiley and Sons, Chichester (1996)
8. Macready, W., Wolpert, D.: Distributed Constrained Optimization with Semicoordinate Transformations. Journal of Operations Research (submitted, 2005)
9. Mamei, M., Vasirani, M., Zambonelli, Z.: Self-Organizing Spatial Shapes in Mobile Particles: The TOTA Approach. Engineering Self-Organising Systems, 138–153 (2004)
10. Modi, P., Shen, W., Tambe, M., Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees. AIJ 161, 149–180 (2005)
11. Ossowski, S.: Constraint-based coordination of autonomous agents. Electronic Notes in Theoretical Computer Science 48, 211–216 (2001)

12. Tumer, K., Wolpert, D.: Coordination in Large Collectives. In: Fifth International Conference on Complex Systems. Perseus Books (2006)
13. Ryder, G.S., Ross, K.: A Probability Collectives Approach to Weighted Clustering Algorithms for Ad Hoc Networks. Communications and Computer Networks, 94–99 (2005)
14. Yokoo, M.: Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems. In: Proceedings of the First International Conference on Principles and Practice of Constraint Programming, pp. 88–102. Springer, Heidelberg (1995)
15. Yokoo, M.: Distributed constraint satisfaction: foundations of cooperation in multiagent systems. Springer, Heidelberg (2001)
16. Waldock, A., Nicholson, D.: Cooperative Decentralised Data Fusion Using Probability Collectives. In: First International Workshop on Agent Technology for Sensor Networks (ATSN 2007), AAMAS 2007 (2007)
17. Weiss, G.: MAAMAW 1997. LNCS, vol. 1237. Springer, Heidelberg (1997)
18. Wolpert, D., Tumer, K.: An introduction to COllective INtelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center (1999)
19. Wolpert, D., Wheeler, K.R., Tumer, K.: General principles of learning-based multiagent systems. In: Etzioni, O., Müller, J.P., Bradshaw, J.M. (eds.) Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS 1999), pp. 77–83. ACM Press, New York (1999)
20. Wolpert, D., Tumer, K.: Optimal payoff functions for members of collectives. Advances in Complex Systems (2001)
21. Wolpert, D.: Information Theory - The Bridge Connecting Bounded Rational Game Theory and Statistical Physics. ArXiv Condensed Matter e-prints (2004)

Tag Mechanisms Evaluated for Coordination in Open Multi-Agent Systems

Isaac Chao[1], Oscar Ardaiz[2], and Ramon Sanguesa[1]

[1] Informatics Department, Polytechnic University of Catalonia, Spain
{ichao,sanguesa}@lsi.upc.edu
[2] Informatics Department, Public University of Navarra, Spain
oscar.ardaiz@unavarra.es

Abstract. Tags are arbitrary social labels carried by agents. When agents interact preferentially with those sharing the same Tag, groups are formed around similar Tags. This property can be used to achieve desired group coordination by evolving agent's Tags through a group selection process. In this paper Tags performance is for the first time compared by simulation with alternative mechanisms for coordinated learning in multi-agent systems populations. We target open systems, hence we do not make costly assumptions on agent capabilities (rational or computational). It is a requirement that coordination strategies prove simple to implement and scalable. We build a simulator incorporating competition and cooperation scenarios modeled as one-shot repeated games between agents. Tags prove to be a very good coordination mechanism in both, cooperation building in competitive scenarios and agent behavior coordination in fully cooperative scenarios.

Keywords: Tags, group selection, multi-agent systems, coordination, prisoner's dilemma, cooperative games.

1 Introduction

Tag-based coordination has already been evaluated for the Iterated Prisoner's Dilemma, (IPD) game and one-shot PD contexts. In this paper we contribute to evolve the state of art in Tag-based coordination in a twofold manner. First, we build a simulator for the TagWorld [1] model for the one-shot PD, and we evaluate the performance of Tags compared with alternative algorithms for multiagent systems (MAS) coordination. Second, we evaluate the Tag mechanism in a different scenario: A pure cooperation game, accomplishing the same comparative study as for the PD. The results here can be useful for researchers aiming to use Tags as a coordination mechanism for engineering purposes.

An open system can be defined as one in which the structure of the system itself is subject to changing. The characteristics of such a system are that its components are not known in advance, can change over time and can consist of highly heterogeneous agents implemented by different people, at different times, with different software tools and techniques [2]. The models that this paper deal with, i.e. one-shot PD and one-shot pure cooperation game, deal with open systems incorporating high uncertainty.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 254–269, 2008.

1.1 Tag Mechanisms

The problem of coordination arises in multi-agent systems (MAS) due to the distributed nature of the control exercised by the agents. Complexity and heterogeneity issues apply in open MAS, increasing enormously the costs of coordination. High dynamicity levels are also common in open systems, with agents entering and leaving the system continuously (e.g. P2P systems), affecting scalability as well. High levels of autonomy required by the agents are also in permanent conflict with coordination mechanisms. Distributed system technologies (e.g. P2P, Grids) evolve more and more into open MAS systems, inheriting this problematic. We provide a coordination mechanism relevant for both selfish and cooperative agents in large, open MAS.

Recent trends in both distributed systems and MAS try to tackle the problem of coordination in MAS from a bottom-up point of view, studying the global properties that emerge from component/agent interaction. Tags can be used as a powerful emergent-coordination mechanism for MAS populations. Holland [3] first proposed the concept of Tags as markings, or social cues, attached to individuals (agents) and observable by others. Agents maintain and modify Tags on themselves and a team is formed by collaborating with agents with the same Tag or that satisfy a specific condition. It is important not to mistake this concept of Tags with the collaborative Tagging phenomena [4], so fashionable nowadays. The Tags we refer to in this paper are arbitrary social labels which do not convey any explicit meaning. Real-life examples of that kind of Tags are gang signals, native tongue and accent, skin color, etc. The Tags referred to in collaborative Tagging conveys meaning and are used to annotate semantically items to simplify further Tag-based search.

Riolo [5] has described a number of Tagging approaches to address the IPD. Riolo outlines basic forms of Tagging: Fixed-bias Tagging, variable-bias Tagging and evolved-bias Tagging. Tags promote the emergence of cooperation between agents even in the single round PD scenario [1]. These techniques are attractive since they don't require centralized or third party reputation systems, the monitoring of neighbor behavior, or the explicit programming of incentives. They can also be used in highly dynamic environments. The results from the single round PD scenario are especially interesting for the engineering of large-scale open distributed systems, since this situation is closer to a real highly dynamic open system, where heterogeneous agents are continuously entering and leaving the system, potentially accounting just for short term interactions.

In a basic Tag model simulation (TagWorld, [1]), each agent maintains a strategy and a Tag (both can be initialized at random). Interaction involves pairs of agents playing a single round of PD. The Tag variable needs to have a quite big space available for variation (a full integer range suffices). This variable has no direct effect on the PD actions selected by the agent but is observable by all other agents. In this setting, a very simple algorithm is applied through a number of rounds: First agents interact preferentially with other agents sharing the same Tag; then agents evolve following an evolutionary algorithm which preferentially reproduces agent's strategies that have collected bigger payoffs. Probabilistic mutation factors on both Tag and action variables are applied. The evolution of the population interacting following the prescribed mechanism precipitates a kind of "group selection" process in which those groups (each group being defined by a Tag) which are more cooperative tend to predominate but still

die out as they are invaded by non cooperative agents. By constantly changing the Tag variable value (by reproduction of those with higher payoffs) the agents produce a dynamic process that leads to high levels of cooperative actions.

Notice that agents remain free to choose the actions "cooperate" or "defeat". They still act in a selfish manner pursuing their own interest. However they commit to apply and respect the Tag algorithm. For more discussion elaborating on the implications of such a requirement, see [6] and [7]. Extensive experimentation varying a number of parameters showed that for a big enough Tag space, high levels of cooperation quickly predominated in the population. Additionally, the fact that the system can recover from a state of total non-cooperative actions to almost total cooperative actions (under conditions of constant mutation) demonstrates high robustness. The Tag-based mechanism produces an efficient, scalable and robust solution based on very simple individual learning methods (modeled as reproduction and mutation).

1.2 Objectives and Motivation for Research

Provided the huge amount of work on MAS learning mechanisms, and the growing literature on Tag mechanisms, this section states the motivation and contributions of this paper. The paper aim is to further investigate the performance and the applicability of emergent coordination mechanisms based on Tags. Most of Tag-based related work is concerned with cooperation building in IPD settings (see all the papers in section 2). A few exceptions target applications of Tags in scenarios others than this. Notably, applications into realistic P2P scenarios are shown in the work by Hales [8], [9], [10]. Also other tentative applications are query-routing and processing for Peer-to-Peer web search [11], preventing free-riding in Grid Virtual Organizations coordination [12], and modeling the dynamics of firms [13].

We identify two important features missing in all this previous work. First, all the scenarios are targeting either cooperation building/free-riding controlling (all the IPD-based studies and the work on P2P systems by Hales) or competitive scenarios where agents have incentives to behave in opposition to the interest of the rest of the agents in the system (the rest of applications). What can we expect about the applicability of Tags into fully cooperative domains? Are Tags useful in those cases? Second, most of the Tag studies have been conducted relatively aside of the related body of work in MAS. Agent systems community has developed a number of learning algorithms which prove simple enough to be used by reactive agents in open systems environments. What knowledge about tag mechanism can we derive by comparing their performance with existent MAS learning algorithms?

The importance of exploring the behavior of Tag mechanism in fully cooperative settings can be realized in practical applications such is Grids. While free-riding is an important concern in P2P systems [8], this issue is not the most relevant in other scenarios such as the huge scientific collaborations in Grids [14]. In these settings, free riding can be controlled through the solid accounting and security mechanisms already implemented. The most relevant issues in this type of systems come from the complex management of workflows, resource management policies and so on, all of which are coordination based tasks, and do not involve necessarily cooperation building.

The results of the simulations show good performance for the Tag mechanism compared with alternatives, in both the cooperation and competition games. The contributions are twofold: We extend the knowledge of Tags in competitive settings (PD) by comparing their performance with other learning algorithms, and we identify a novel potential for the Tag mechanism, namely improved scalability of fully cooperative MAS settings. This opens a new path of applications of the Tag mechanisms, improving the scalability of cooperative learning agents, just by structuring the population of agents in groups and evolving these groups following the Tag algorithm (that is applying a process of group selection).

The rest of the paper structures as follows. Section 2 evaluates related work, relating its contributions to this paper. Section 3 details experimentation settings, the Tag mechanism model and the alternative learning mechanisms simulated. Section 4 shows the core results: presents the experimental setup and the performance results on Tags compared with the alternative coordination mechanisms in both competitive and cooperative scenarios. A final subsection discussing the results and its applications is provided. Section 5 concludes the paper and outlines future work.

2 Related Work

In addition to the seminal research by Holland [3], by Riolo [5] in the IPD setting, and by Hales [1] in the one-shot PD setting, several recent papers have studied in-depth different aspects of Tag models. We present a summary of the most important conclusions and compare their contribution to ours.

In [15], the emergence of cooperation in simple Tag models incorporating IPD is studied by simulation. The results signal the importance of population viscosity (understood as static populations) in promoting cooperation between agents. Their simulation also proves that high Tag spaces are required for the emergence of cooperation. Although the number of Tag values used by the agents fall significantly after the initial generations, the large number of Tags in the beginning is essential. The Tag mechanism has the ability to marginalize non cooperative behaviors over the initial populations

In the models from [6], it is confirmed by simulation that cooperation in Tag models is evolved based on fitness if sufficient number of new groups are created via mutation. There has to be enough groups such that the rate of destruction via invasion by defectors is less than the formation of new groups by mutation. More interestingly, they build a partial theoretical characterization of this model, throwing the conclusion that Tag systems are merely promoting mimicry, rather than cooperation. In order to test this hypothesis, they build a model with agents playing a pure anti-coordination game instead of the PD. They conclude that when cooperation requires complementary agents Tags do not lead to cooperation. However, simulations by Hales [16] and Edmonds [17] show that some level of specialization can be derived between agents using Tags.

The two papers evaluated so far give a comprehensive evaluation on basic Tag mechanism behavior in PD settings (as pioneered by Riolo and Hales) but contrary to our paper, they do not provide any comparison with alternative learning mechanisms. Also they focus on competitive scenarios and do not target fully cooperative games.

Research in [18] extends the use of Tags to interaction between groups, and not just to segment the population on groups of interacting agents as in previous models. The results of the simulations show that Tags incorporate some level of reciprocity between groups. In [19] they present a Tag model incorporating sexual reproduction (recombination) of agents. Analyzing this model they find occasional formation of very stable cooperative societies, able to resist invasion of mimics (defecting agents with the Tag of a cooperative agent). Both models present extensions of Tag mechanisms, but no comparison with alternatives is present.

Perhaps the work which can be considered closer to this paper is presented in [20]. Here, a wide comparison between many interaction-biasing processes is performed, including several topology-based, others based on random networks of neighbors and also Tags. The extensive simulation includes strategy variations and adaptation process variations for each of the interaction-biasing processes. As in our case, they approach the study of the performance of Tags under many different settings and compared with many other mechanisms. The important result they achieve is that context-preservation, topologically-based or not, is essential in promoting cooperation. Tags are shown to perform in-between full context preservation topological-based interaction processes and no context preservation process. In the Tag interaction process neighbors will tend to be chosen from a pool of like-Tagged agents, which is much smaller sized than the whole population of agents. This leads to an increasing probability of context preservation. This confirms the hypothesis by Howley [15] on viscosity requirements in Tag mechanisms, coming back to the findings on biological population's viscosity by Hamilton [21].

However, there are several important differences between the approach in [20] and the one in this paper. First, in their simulations they use IDP whereas we use one-shot PD. As we mentioned earlier, this change is motivated by our intention to approach fully open systems. Also, in their study, performance comparison is targeted towards interaction processes analysis, rather than coordination as in our case. I.e., they approach Tags at the level of agents, acting as an interaction mechanism, while we approach Tags as a coordination mechanism for the whole system. This makes both comparative examinations complementary.

3 System Model and Learning Mechanisms

3.1 System Model: Cooperation and Competition Models

We use two fundamental games of game theory in order to represent the two basic scenarios: These are cooperation, where roughly individual and social welfare match, and competition of conflicting interests, where this is not necessarily the case (e.g. social dilemmas). The pure cooperation game and the prisoner's dilemma respectively abstract these scenarios.

The PD (Table 1) is a type of non-zero-sum game in which two players try to get rewards by cooperating with or betraying the other player. In the PD, cooperation is strictly dominated by defection (i.e., betraying one's partner). Since in any situation defection is more beneficial than cooperation, all rational players will defect (Nash Equilibrium). The unique Nash equilibrium for this game is a Pareto-suboptimal

Table 1. PD Game

	P2 cooperates	P2 defects
P1 cooperates	R,R	S,T
P1 defects	T,S	P,P

Table 2. Pure Cooperation Game

	P2 action 1	P2 action 2
P1 action 1	A,A	B,B
P1 action 2	C,C	D,D

solution—that is, rational choice leads the two players to defect even though each player's individual reward would be greater if they both decide to cooperate. The challenge is to provide incentives in the repeated game for the agents to achieve the Pareto optimal solution maximizing population welfare, mutual cooperation. Let T stand for Temptation to defect, R for Reward for mutual cooperation, P for Punishment for mutual defection and S for Sucker's payoff. The following inequality must hold in a PD: $T > R > P > S$. If the game is iterated or repeated, the mutual cooperation total payment must exceed the temptation total payment: $2R > T + S$.

The pure cooperation game is symmetric, two players, two strategies, with payoff matrix as given in Table 2. In the coordination game the following holds: A>C and D>B. Players in the game must agree on one of the two strategies in order to receive a high payoff. If the players do not agree, they receive a lower payoff. This game represents a common scenario in MAS systems when many agents' goals are to be aligned, leading to very suboptimal outcomes when this is not the case.

3.2 Tag Mechanism Model

The proposed Tag model (see algorithm in figure 1) is close to TagWorld [1], except a few details: We bootstrap the agents randomly into a number of groups (identified by a Tag) from the beginning, as opposed to Hales model where agents begin each one with a randomly given Tag. This initial bootstrapping is motivated to model real scenarios where organizations are already in some specific configuration; as expected, we did not find any impact for the two games analyzed of this initial bootstrapping. The second difference is that we explicitly forbid agent operation outside the group. In the case of not finding a suitable mate in its group the agent just skips operation in this round and goes directly to the evolution phase. This is motivated by making the group as the scope of agent's operation. This should not make a big difference since isolated agents tend to migrate to other groups looking for bigger payoffs. The last difference is that we mutate just Tags and do not introduce mutation in strategies. This is unnecessary for the emergence of cooperation/coordination (contrarily to Tag mutation). Equivalent effect to action mutation can be attained by introducing noise in the game play (e.g. consider that the action is wrongly interpreted in 1% of the cases, this will be equivalent to a 1% of mutation in action). We refer to Hales [1] for an analysis of action mutation impact and the robustness of the Tag mechanisms to it.

```
Bootstrap agents in groups
LOOP a number of rounds
    LOOP each group
        LOOP each agent in the group (operation phase)
        Interact with another agent from the group (i.e. same Tag)
            Collect Payoff
        ENDLOOP
    ENDLOOP
    LOOP each agent in the population (evolution phase)
            Select partner agents in the population
        If partner outperforms agent
            Then copy partner Tag (migrate to its group) and action
        Mutate: agent applies probabilistic Tag mutation
    ENDLOOP
ENDLOOP
```

Fig. 1. Proposed Tag algorithm

The interaction involves Tag-biased mate selection (i.e. within the group) and bilateral playing of the game. The corresponding payoff is collected by the initiator agent. The evolution phase is ruled by evolutionary learning, replication of the fittest. This is common in all Tag models from Riolo to Hales and others. Summarizing, the Tag mechanism model is very close to the one by Hales, except several details. The difference comes from the type of interactions; we will test the performance in pure cooperation scenarios, and not only in social dilemma games such as PD.

An implicit assumption of the Tag mechanism is that agents are capable of comparing utilities and relating them to actions. This is the case for the proposed games given that the agents know the payoff matrix. Different applications than the games tested here may require adequate services to assess/compare utilities. Another assumption is the existence of a reliable discovery mechanism that allows agents to locate other agents sharing the same Tag (i.e. belonging to the same group). Also, agents are able to communicate exchanging information. Scalability issues arising in practical applications from this assumption can be mitigated by sorting agents in groups following their Tags, reducing the search space.

3.3 MAS Learning Algorithms Selection

In order to achieve scalability in opens systems, we limit the type of the coordination mechanism allowed, to those meeting a set of conditions and practical issues: Being simple to implement (i.e. deploy and use) in real systems; not imposing computational or other expensive requirements on the agents; not requiring a central coordination component (i.e. be self-organizing). Properties are expected to emerge at the system level from individual agent's interaction. We exclude team learning and coalition

formation literature from the set of mechanisms selected. Traditionally, most of the coalition formation algorithms are rather theoretical models based on deliberative agents. Limitations of these algorithms that render them inapplicable to large scale distributed systems are a high computational complexity, and unrealistic assumptions regarding the availability of information [22]. Some novel coalition formation mechanism could be incorporated in a simulation with more different scenarios [23] [24], though this is left for future work.

From the set of possible MAS mechanisms to include in the simulation complying with the characteristics above, we have selected two important agent strategies emerging from game theory IPD tournaments, namely TFT and the Pavlov strategy (also called win stay, loose shift). We have introduced a simple Reinforcement Learning (RL) algorithm based on the extensions on the Pavlov strategy. Still in the camp of RL, we consider a basic Q-Learning algorithm. The last learning mechanism we have incorporated is a simple genetic algorithm for the whole population, with mimicry of the fittest alternative. It is important to notice that the proposed Tag algorithm incorporates the same evolutionary algorithm, with the only difference of splitting the population in groups. We use the following alternative learning mechanisms for the comparison:

- Generalized TFT, with random initialization of the action. This differs from the original TFT definition [25], but this is normal if we consider that the strategy is applied to the whole population, which would render a trivial game by starting all agents from cooperation. For the PD scenario TFT means a player defect just if the previous partner (the previous agent with whom he played) did so. Here TFT is applied to successive one-shot interactions with different agents, which is a big difference from typical TFT applied in IPD tournaments: There, each interaction partner is maintained for a definite period before switching to a new partner. The purpose is to show how the population effectively converges to coordinated outcomes under this simple strategy, even in one-shot interactions. For the pure cooperation game, TFT means to blindly copy previous partner agents, in the hope of having the whole population converging via this heuristic. Though this is of improbable success, we keep all learning mechanisms in both games to render some symmetry in the comparison.

- WSLS (Win stay, loose shift) or Pavlov strategy [26]. Here an agent defects only if both players do not agree on the previous move. It is a kind of TFT complement. The idea behind WSLS is to allow agents to use their previously-experienced utilities in order to decide next action, rather than partner's previous moves as it is done TFT. This makes the agents more reactive to their own experiences. The intuition is that emergence of coordination will be harder in one-shot scenarios than in IPD, since agents will have harder time relating utility to actions leading to payoff increases.

- RL (Basic Reinforcement Learning algorithm). Reinforcement Learning has been widely studied in agent theory (mostly in static environments) and in MAS settings. However the body of work in multi-agent RL is still small, contrasting with the literature on single-agent learning, as well as the literature on learning in game theory [27]. It has been problematic extending convergence results to stochastic games. Consider for example the algorithm in [28]: The RL formula for the agent internal state (Equation 1) can be explained as follows: We have for the agent an internal

state, h, representing its satisfaction level. The learning player plays C (cooperation) when h>0, otherwise D (defection). If C is played and the resulting score f is larger than s (a fixed constant aspiration level), h increases. If f is smaller than s, then h decreases. Conversely, successful D decreases h and vice-versa. The constant aspiration level s is taken in the interval $1 \leq s \leq 3$. Specifically, we set s to 2. The interested reader may see more details about how this relates with the payoffs we use for the PD in [28]. As for the pure cooperation game, we expect a good performance of this algorithm.

$$\Delta h = a \cdot \mathrm{sgn}(f - s) \cdot \mathrm{sgn}(h) \tag{1}$$

- Q-Learning. The Q-learning [29] is a form of RL algorithm that does not need a model of its environment and can be used on-line. Its convergence conditions have been widely studied, but convergence results in MAS have been always hard to achieve, rendering most of the advances to plausible heuristics [30]. We chose the simpler Q-learning available with typical values for the parameters, including e-greedy selection. Again, a good performance for the coordination game is expected. For the PD, we have precedent studies showing good results for the IPD [31]. However the performance in one-shot PD might differ from the one in IPD scenarios. We use the formula given in Equation 2 for updating the Q-value. In this formula, α is the learning rate: The bigger the learning rate the more important is the impact of environment reinforcement. We set up a learning rate of 0.5. As for the Q-value selection, we use an e-greedy policy with a probability of 0.1 of exploring randomly and a probability of 0.9 of exploiting the highest Q-value.

$$Qi,t+1 \leftarrow Qi,t + \alpha(r - Qi,t) \tag{2}$$

- Evolutionary Learning (Evo). Applying the most basic evolutionary learning, agents reproduce asexually copying the action played by the fittest partner they find. In our setting each agent chooses a partner randomly and copies its action if the agent outperforms him. In the same way that previous strategies (TFT and WSLS) and learning from reward mechanisms (basic RL, Q-learning) are used to decide and evolve the agent's next action, the evolutionary learning mechanisms (Evo and Tags) are used to derive next actions. What gets copied are agents actions (e.g. cooperate or defect for the PD, 0 or 1 for the pure cooperation game) and not strategies (e.g. RL strategy, TFT strategy, etc). Since the evolutionary algorithm is also used in the Tag mechanism itself, the comparison can be used to further evaluate which building blocks from typical Tag mechanisms are promoting cooperation/coordination and how. For a full theoretical evaluation of this type evolutionary algorithm in a varied range of mutation levels, the interested reader may see [32].

4 Experimental Results and Discussion

Given a population of N agents playing repeatedly one-shot PD and the pure cooperation game, we want to compare the performance of several coordination mechanisms in order to maximize social welfare.

4.1 Experimental Setup

We build a simulator in java enabling the implementation of different agent coordination protocols. Relevant parameters for the simulator are shown in Table 3. We fix a population of 100 agents, a representative size big enough to consider emergence in Tag-based coordination. Each experiment run is composed of 500 rounds to be able to average results over long runs. Tag mutation rate is fixed in the comparison with alternative MAS coordination mechanisms to m=0.01 (cf. Table 3), a normal value in TagWorld, high enough to promote variability without fully destabilizing the system. This is a typical value widely used in previous studies.

Table 3. Experimental parameters

NAME	VALUE
NUMBER OF ROUNDS	500 (fixed)
AGENT POPULATION	100 (fixed)
TAG MUTATION PROBAB m	0.001, 0.01, 0.1

Agents are initialized with random actions in all cases, and given an initial Tag (in the case of the Tag mechanism). We instantiate the matrices for the PD and coordination games as shown in Table 4. The payoffs for the pure cooperation game are the canonical ones.

Table 4. Payoff Matrix instantiation

PD	Cooperate	Defect	//	COORD	Action 1	Action 2
Cooperate	3,3	1,4	//	Action 1	1	-1
Defect	4,1	1,1	//	Action 2	-1	1

4.2 Performance Comparison in the PD Game

For the PD game we measure two scores: Cooperation Level, the population percentage cooperating; varies on the range 0 to 1, and Average Utility: Social Welfare; the population average utility. The bounds are 1 (Sucker payoff) and 4 (Temptation payoff). We are interested in the long term behavior of the Tag mechanism compared to the rest of mechanisms. We calculate on each round the average of the property (Cooperation Level or Average Utility) over the total number of rounds (500). We repeat the experiment for each parameters configuration 10 times and calculate the average with standard deviation as result of the experiment for each setting. The results are shown in Table 5. We provide an additional graphical display of a sample experiment run in figure 2.

Figure 2 shows, in the long run, similar performance of TFT and Tags in emerging total cooperation in the PD, but several rounds before in Tag-based models. The same stable pattern holds in rounds 200-500m which is not shown here. From the long run tabular results (Table5) we see a higher variance on the TFT mechanism in reaching

Table 5. Performance comparison for the PD. Mean and standard deviation over 10 experiment runs

PD GAME MECHANISM	COOP LEVEL MEAN	COOP LEVEL SD	/////	PD GAME MECHANISM	AV UTILITY MEAN	AV UTILITY SD
TFT	0,87	± 0,32	/////	TFT	2,72	± 0,42
WSLS	0,5	± 0,01	/////	WSLS	2,23	± 0,01
RL	0,20	± 0,23	/////	RL	1,72	± 0,48
QL	0,52	± 0,01	/////	QL	2,37	± 0,01
EVO	0,02	±0,01	/////	EVO	1,27	±0,03
TAGS (m=0.01)	0,98	± 0,01	/////	TAGS (m=0.01)	2,84	± 0,02

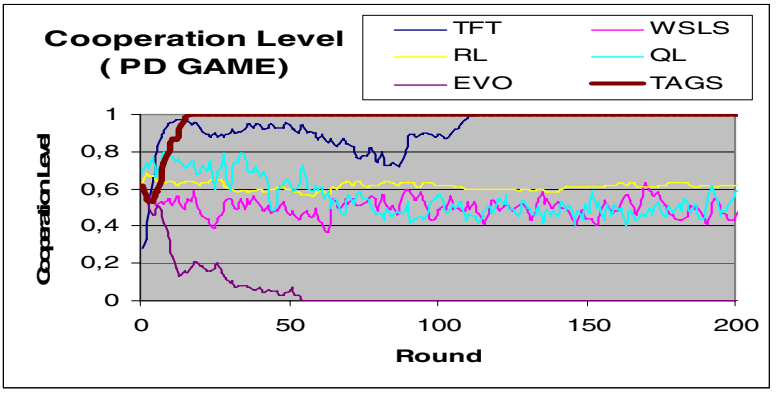


Fig. 2. Performance results for a PD Game

cooperation, i.e. more unstable. Pavlov strategy (WSLS) is just able to maintain the initial random distribution of strategies, and never converges to any pure cooperation or defection. The simple RL mechanism performs badly; the one-shot PD is a harder scenario than the IPD for coordination. The Q-Learning mechanism maintains a stable equilibrium between cooperative and non cooperative agents but is not able to promote further cooperation. From the tabular results in table 5 we can see an important difference on the variance, much higher in RL, proving a better stability on the Q-Learning. The evolutionary mechanism is not able to sustain cooperation. This confirms that niches formed by dividing the population in groups sharing Tag are essential in promoting cooperation (the context preservation referred in [20]).

4.3 Performance Comparison in the Pure Cooperation Game

For the Coordination Game we measure the Average Utility or Social Welfare. The bounds are 1(total coordination) and -1 (total uncoordinated behavior).

Table 6. Performance comparison for the Coordination Game. Mean and standard deviation over 10 experiment runs.

COORD GAME MECHANISM	AV UTILITY MEAN	AV UTILITY SD
TFT	-0,01	± 0,01
WSLS	-0,01	± 0,01
RL	-0,00	± 0,01
QL	0,05	± 0,01
EVO	0,94	±0,01
TAGS (m=0.01)	0,95	± 0,01

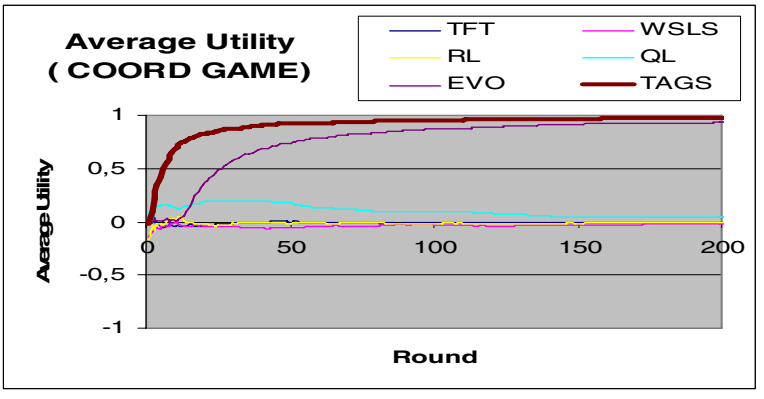


Fig. 3. Performance results for the Coordination Game

Considering the pure cooperation game (Table 6, figure 3), TFT and WSLS are not helpful (as expected). The RL does not achieve any important improvement. The more elaborate Q-Learning algorithm is able to evolve a small level of coordination. This result is unexpected since reinforcement learning agents should achieve good performance in a so simple coordination game. However, again we recall multiagent RL literature on the controversy of convergence results in multiagent settings [33]. Our simulation shows two (basic) reinforcement learning mechanisms not converging on the PD and pure cooperation game scenarios. In contrast, Tags are able to coordinate the population. The evolutionary algorithm also performs very well. We conclude that it is the evolutionary aspect from Tag mechanisms which is mostly provoking the convergence of actions in fully cooperative domains.

4.4 Discussion and Applications

We summarize here the most important conclusions of this comparative study. First, the Tag mechanism shows a very good performance in both scenarios: The PD and the pure cooperation game. This confirms in the broader scope of previous work in MAS, Riolo, Hales and others (see section 2) and extends the state of the art of the Tag mechanisms applied to fully cooperative domains. This expands Tag mechanisms from just social dilemmas and competitive settings to a full range of MAS applications,

bridging its application to both competitive and cooperative multiagent learning fields [34]. Second, we show how simple MAS learning mechanisms have worst performance in both competitive and cooperative scenarios, in the repeated one-shot games settings. It has been surprisingly found a bad performance of the reinforcement learning agents (simple RL and Q-Learning) in both games; especially surprising in the case of the pure cooperation game. We propose two plausible reasons for this to happen: First the problems of RL with convergence in MAS settings anticipated in literature [33]. Second the fact that interactions are not iterated, rendering a much more unpredictable context for agents interactions. In such environment, simple model-free reinforcement learning agents get a hard time coordinating actions.

Tag mechanisms are a very promising mechanism for evolving swarms of agents into optimized outcomes. The applications of these results are varied in open MAS, in demanding scenarios where alternative learning mechanisms might fail or prove inapplicable due to computational requirements limitations. The most compelling are those envisaged for coordination purposes in Service Oriented Architectures and Next Generation Computational Grids [14]. In this latter scenario, the management of first level system entities, Virtual Organizations (VOs), can be mapped directly to the concept of a group (identified by a Tag). In fact, most of the slight variations we introduced on the TagWorld model (see section 3.2) target the full alignment with the original VO concept in Grids. A research agenda has been outlined in [12], and partially addressed the form of an application to automatic alignment of resource management policies in VOs [35].

5 Conclusions and Future Work

In this paper Tags are for the first time compared by simulation with alternative mechanisms for coordinated learning in MAS populations. We target open MAS, hence we do not make costly assumptions on agent rational or computational capabilities. It is a requirement for us that coordination strategies prove simple and scalable. Tags are simple, requiring from the agents to maintain a marker visible to the rest of the agents. They show equal or better performance in the two games than any other of the mechanisms tested. Tags still remain loosely coupled with the system, which enables for combination with other MAS coordination mechanisms. An example of this complementarily can be found in [36], showing a mechanism which uses Tags optimizing decentralized Grid markets.

As we have reviewed in section 2, many related work has evaluated Tag mechanism internals for the IPD. Direct relation of the emergence of cooperation with mechanism parameters such as Tag space length and mutation rate is clearly identified and analyzed. However no prior study has evaluated how Tag mechanism performs in one-shot PD, compared to the alternative MAS learning algorithms. With the experiments from section 4, we have addressed this lack of comparability. We have found that Tags effectively work as indirect reciprocity enabler. Most of the algorithms had a hard time stabilizing their learning in the one-shot PD setting. Reciprocity built on repeated interaction with a same partner (as for TFT in IDP settings) or static contexts (as in single agent reinforcement learning) reveal crucial factors for the rest of the mechanisms in order to achieve accurate learning. The basic

strength of Tag mechanism is that it shows robust behavior in a demanding setting where other single MAS coordination mechanisms fail. Apart from reputation mechanisms, there are very few mechanisms addressing indirect reciprocity. Tags generate indirect reciprocity without the need of complex, system dependent reputation management, and perform well compared with alternatives. A partially theoretical analysis of all the mechanisms enabling indirect reciprocity (including the group selection process behind Tag mechanism) can be found in [37].

As for the pure cooperation game, Tags achieve a good performance while the rest of the mechanisms again have problems coping with short-term interaction. The evolutionary mechanism is the exception, achieving a comparable performance. It is the evolutionary learning present in the Tag mechanism the main driver of performance increase in this scenario. An important open issue for state-of-the-art Tag models is how to achieve coordination between complementary policies, which is difficult since Tags promote basically mimicry [6]. A solution to this issue may involve changes in the evolution of agent strategies when entering new groups, or alternatively, trying more elaborate Tag similarity measures for interaction biasing.

Future work will comprise three basic areas. The first area is Tag mechanism extension to multiple Tags per agent and variations on the learning, using alternatives to evolutionary learning. Complementing the simulation approach with theoretical analysis, such as proposed in [38] can elucidate better simulation scenarios, leading to improved accuracy. The second area consists on extending the comparison to more general coordination and organizational mechanisms, such as decentralized markets [39], and explore combinations between them (Tags are highly modular hence bear easy integration with existent mechanisms). Scalable coalition-formation mechanisms [22] could also be included in this area of research. A third area of improvement is on realistic models, beyond coordination games. In such models payoffs are based on real tasks execution modeling [40] and not in an abstract matrix. Deployment of Tags in a real prototype can lead to a performance evaluation of Tags in real networks settings.

References

1. Hales, D. (2000) Cooperation without Space or Memory: Tags, Groups and the Prisoner's Dilemma. In Moss, S., Davidsson, P. (Eds.) Multi-Agent-Based
2. Sycara, K.: Multiagent Systems. AI Magazine 10(2), 79–93 (1998)
3. J. Holland. The effects of labels (Tags) on social interactions. Working Paper Santa Fe Institute 93-10-064 (1993)
4. Golder, S.A., Huberman, B.A.: The Structure of Collaborative Tagging Systems. Information Dynamics Lab, HP Labs (Visited November 24, 2005)
5. Riolo, R.: The efects of Tag-mediated selection of partners in evolving populations playing the iterated prisoners dilemma. Nature 414, 441–443 (2000)
6. McDonald, A., Sen, S.: The Success and Failure of Tag Mediated Evolution of Cooperation. In: Tuyls, K., 't Hoen, P.J., Verbeeck, K., Sen, S. (eds.) LAMAS 2005. LNCS (LNAI), vol. 3898, pp. 155–164. Springer, Heidelberg (2006)
7. Arteconi, S., Hales, D., Babaoglu, O.: Greedy Cheating Liars and the Fools Who Believe Them. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335. Springer, Heidelberg (2007)

8. Hales, D.: From Selfish Nodes to Cooperative Networks – Emergent Link-based Incentives in Peer-to-Peer Networks. In: Proceedings of The Fourth IEEE International Conference on Peer-to-Peer Computing (p2p2004), Zurich, Switzerland, August 25-27, 2004. IEEE Computer Society Press, Los Alamitos (2004)
9. Hales, D., Patarin, S.: Feature: Computational Sociology for Systems In the Wild: The Case of BitTorrent. IEEE Distributed Systems Online 6(7) (2005)
10. Hales, D., Babaoglu, O.: Towards Automatic Social Bootstrapping of Peer-to-Peer Protocols. ACM SIGOPS Operating Systems Review (Special Issue on Self-Organizing Systems) 40(3) (July 2006)
11. Weikum, G., Triantafillou, P., Hales, D., Schindelhauer, C.: Towards Self-Organizing Query Routing and Processing for Peer-to-Peer WebSearch. In: Proceedings of the European Conference on Complex Systems (ECCS 2005), Paris, France, November 14, 2005. i6doc, Belgium (2005) (in press)
12. Chao, I., Ardaiz, O., Sanguesa, R.: Tag Mechanisms Applied to Open Grid Virtual Organizations Management. In: Anthony, R., Butler, A., Ibrahim, M., Eymann, T., Veit, D.J. (eds.) Proceedings of the Joint Smart Grid Technologies (SGT) and Engineering Emergence for Autonomic Systems (EEAS) Workshop, Dublin, Ireland, pp. 22–29 (2006)
13. Mollona, E., Hales, D.: Modeling Firm Skill-Set Dynamics as a Complex System. In: Proceedings of the European Conference on Complex Systems (ECCS 2005), Paris, France, November 14, i6doc, Belgium (in press, 2005)
14. Next Generation Grids Expert Group Report 3, Future for European Grids: GRIDs and Service Oriented Knowledge Utilities
15. Howley, E., O'Riordan, C.: The Emergence of Cooperation among Agents using Simple Fixed Bias Tagging. In: IEEE Congress on Evolutionary Computation, September 2-5 (2005)
16. Hales, D.: The Evolution of Specialization in Groups. In: Lindemann, G., Moldt, D., Paolucci, M. (eds.) RASTA 2002. LNCS (LNAI), vol. 2934. Springer, Heidelberg (2004)
17. The Emergence of Symbiotic Groups Resulting from Skill-Differentiation and Tags, Bruce Edmonds. JASSS 9(1), January 31(2006)
18. Zohar, A., Rosenschein, J.S.: Using Tags to Evolve Trust and Cooperation Between Groups. In: The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, July 2005, The Netherlands, pp. 1199–1200 (2005)
19. Floortje Alkemade, D.D.B., van Bragt, J.A.: La Poutré: Stabilization of Tag-mediated interaction by sexual reproduction in an evolutionary agent system. Inf. Sci. 170(1), 101–119 (2005)
20. Cohen, M., Riolo, R., Axelrod, R.: The emergence of social organization in the Prisoner's Dilemma: how context-preservation and other factors promote cooperation Santa Fe Institute Working Paper 99-01-002 (1999)
21. Hamilton, W.D.: Man and Beast: Comparitive Social Behaviour. In: Eisenberg, J.F., Dillon, W.S. (eds.). Smithsonian Press, Washington (1971)
22. Coalition Formation: Towards Feasible Solutions. Fundamenta Informaticae 63(2-3), 107–124 (2004)
23. Lerman, K., Shehory, O.: Coalition formation for large-scale electronic markets. In: Proc. of ICMAS 2000, Boston, MA, pp. 167–174 (2000)
24. Kraus, S., Shehory, O., Taase, G.: Coalition formation with uncertain heterogeneous information. In: Proc. Of AAMAS 2003, Melbourne, Australia, pp. 1–8 (2003)
25. Axelrod, R.: The Evolution of Cooperation. Science 211(4489), 1390–1396 (1981)
26. Nowak, M., Sigmund, K.: A strategy of winstay,lose-shift that outperforms tit-for-tat in the prisoner'sdilemma game. Nature 364, 56–58 (1993)

27. Shoham, Y., Grenager, T., Powers, R.: Multi-agent reinforcement learning: A critical survey. Tech.rep., Stanford University (2003)
28. Wakano, J.Y., Yamamura, N.: A Simple Learning Strategy that Realizes Robust Cooperation Better than Pavlov in Iterated Prisoners' Dilemma. J. Ethology 19, 9–15 (2001)
29. Watkins, C.: Learning from Delayed Rewards, Thesis, University of Cambidge,England (1989)
30. de Cote, J.E.M., Lazaric, A., Restelli, M.: Learning to cooperate in multi-agent social dilemmas. In: AAMAS 2006, pp. 783–785 (2006)
31. Sandholm, T., Lesser, V.: Coalitions Among Computationally Bounded Agents. Artificial Intelligence, special issue on principles of multiagent systems 94(1) (1997)
32. Willensdorfer, M., Nowak, M.A.: Mutation in evolutionary games can increase average fitness at equilibrium. J. theor. Biol. 237, 355–362 (2005)
33. Bowling, M.: Convergence problems of general-sum multia-gent reinforcement learning. In: Proceedings of the Seventeenth International, Conference on Machine Learning, pp. 89–94 (2000)
34. Durfee, E.H., Rosenschein, J.S.: Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples. In: Proc. 13th Int'l Distributed Artificial Intelligence Workshop, pp. 94–104 (1994)
35. Chao, I., Ardaiz, O., Sanguesa, R.: A Group Selection Pattern for agent-based Virtual Organizations coordination in Grids. In: International Conference on Grid computing, High-Performance and Distributed Applications (GADA 2007), Vilamoura, Algarve, Portugal, November 29- 30 (2007)
36. Chao, I., Ardaiz, O., Sanguesa, R.: A Group Selection Pattern Optimizing Job Scheduling in Decentralized Grid Markets. Poster accepted to the International Conference on Grid computing, High-Performance and Distributed Applications (GADA 2007), Vilamoura, Algarve, Portugal, November 29 - 30 (2007)
37. Nowak, M.A.: Five rules for the evolution of cooperation. Science 314, 1560–1563 (2006)
38. Gotts, N.M., Polhill, J.G., Law, A.N.R.: Agent-based simulation in the study of social dilemmas. Artificial Intelligence Review 19, 3–92 (2003)
39. Eymann, T., Reinicke, M., Streitberger, W., Rana, O., Joita, L., Neumann, D., Schnizler, B., Veit, D., Ardaiz, O., Chacin, P., Chao, I., FreiTag, F., Navarro, L., Catalano, M., Gallegati, M., Giulioni, G., Schiaffino, R.C., Zini, F.: Catallaxy-based Grid Markets. In: Veit, D.J., Eymann, T., Jennings, N.R., Müller, J.P. (eds.) Multiagent and Grid Systems Issue: Smart Grid Technologies & Market Models, vol. 1(4), pp. 297–307 (2005)
40. Galstyan, A., Czajkowski, K., Lerman, K.: Resource Allocation in the Grid with Learning Agents. Journal of Grid Computing 3(1–2), 91–100 (2005)

Toward a Probabilistic Model of Trust in Agent Societies

Federico Bergenti

Dipartimento di Matematica
Università degli Studi di Parma
Viale G. P. Usberti, 53/A – 43100 Parma, Italy
federico.bergenti@unipr.it

Abstract. The literature about trust in societies of agents collects a huge number of works that analyse almost any facets of this concept from nearly every point of view. Nevertheless, an accepted and stable formal model of trust in agent societies is lacking. In this paper, we address this remarkable flaw of the current research by introducing a probabilistic model of trust capable of capturing two-party interactions, either direct of mediated by a Guarantor. Some interesting properties of this model are demonstrated and the final result of this work is an estimation (upper-bound) of the improvements that we expect from the inclusion of a Guarantor in a two-party interaction. In details, after an introductory section, Section 2 provides the foundations of our model and quantifies the increment of the utility that agents perceive because of the mediation of a Guarantor. Then, Section 3 deals with the decision-making strategies of rational agents and it shows a worst-case specialization of our model that justifies why agents are more likely choosing Guarantor-mediated interactions. Section 4 describes the overall results of this work in terms of bounds and evaluation of performances of the effects of mediation in interactions. Finally, Section 5 summarizes the main outcome of this work and outline some future lines of development.

1 Introduction

Interaction is a key feature of agenthood (maybe "the" key feature) and secure, trusted and privacy-aware interactions are what we truly want from real-world agent societies [4]. While it is easy to identify a minimum set of requirements capable of providing guarantees of security in multi-party interactions, e.g., authorization and authentication, we are not yet ready to identify similar requirements for trusted and privacy-aware interactions.

This work is along the lines of the research that is trying to identify a set of abstractions and mechanisms to guarantee trust and privacy-awareness in multi-agent interactions. In particular, the objective of this work is to develop a quantitative and probabilistic model of trust in order to show a sound proof of the convenience of Guarantor-mediated interactions over direct interactions. This objective is addressed taking into account a toy scenario that counts agent

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 270–283, 2008.

X and agent Y only (two-party interaction). X is interested in signing a contract with Y and it is in the process of deciding whether to do it directly or through the mediation of a middleman, the Guarantor G. We take a rational standpoint and we assume that X discriminates between direct and mediated interaction on the basis of a utility function. Moreover, we take an incomplete information assumption and we say that X cannot take a fully-informed decision; rather it has to deal with a risky situation, which immediately turns our model into a probabilistic one.

The main results of the study of this toy scenario are based on a worst-case analysis of a much more general model and they provide a sound proof of why rational agents are more likely choosing Guarantor-mediated interactions rather then direct interactions.

This paper is organized as follows: next section provides the foundations of our model and it quantifies the increment of the utility that agents perceive after the inclusion of a Guarantor in a two-party interaction. Section 3 deals with the decision-making strategies of rational agents and it quantitatively shows why agents are more likely choosing Guarantor-mediated interactions. Section 4 describes the result of this work in terms of bounds and evaluation of performances of the effects of mediation in interactions. Finally, Section 5 summarizes the overall results of this work and outline some future lines of development.

2 A Model of Guarantor-Mediated Interactions

This section presents a set of abstractions and accounts for their relationships in order to setup a probabilistic model of interactions between agent X, agent Y and (possibly) Guarantor G. It is worth noting that this model is symmetrical for X and Y.

2.1 Abstractions

A very basic assumption that we take in the discussion of our model is that, from the point of view of security, trust and privacy, we can always reduce a two-party interaction to the act of signing of a contract. Therefore, from now on, we always refer to the joint act of signing a contract as a means to study any other form of two-party interaction.

Trust. The problem of providing a quantitative definition of trust in societies of rational agents has been addressed in many different ways [14]. While we recognize the importance of cognitive models of trust, e.g., [5], we date back to the abstract and coarse-grained definition of trust given in [9] to come to a probabilistic interpretation this notion.

In particular, if *"Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends,"* it is quite reasonable to model trust as an estimation of the probability by which B will perform the target action. Many factors contribute to this estimation [11,13]; nonetheless we prefer to discard all these factors and we

adopt a blackbox approach in which we model trust as a random variable $\mathbf{t}$ in an interval $[t_{min}, t_{max}]$.

The only assumption that we take in our model is that we require such an estimation to be performed by a rational agent A with some reasonable amount of information regarding B and its intentions of performing the action. This guarantees that the real probability of B performing the action lays in $[t_{min}, t_{max}]$, with both t_{min} and t_{max} reasonably strict around it.

Our model of two-party interactions relies on the following quantities, where X and Y are agents and c is a contract:

- $p_{c,X}$: the probability that X would carry out successfully all the obligations stated in c.
- $t_{c,X,Y}$: the level of trust X has in Y with respect to c, i.e., an estimation of $p_{c,Y}$ from the point of view of X.

Since trust expresses the estimation of a probability, it is clear that t_{min} and t_{max} are both between zero and one. The assumption $t_{max} \geq t_{min}$ is not restrictive.

Contract. The study of all different forms of contract is subject of a large literature and even restricting to the types of contract that we normally consider in societies of agents [3], the diversity of possibilities is impressive. We acknowledge this literature, but for the sake of simplicity and for the need of quantitative tractability, we stick on a very simple model of contract. This model involves only two signers, X and Y, and it is totally described by two triples: each signer knows only one of the two triples.

From the point of view of agent X (but the notation is symmetrical for Y), a contract c is described by a triple, that we call *subjective evaluation*, that contains:

- A *reward* $R_{c,X}$ that agent X receives upon success of contract c;
- An *investment* $I_{c,X}$ that agent X makes in contract c, i.e., a certain assured value that it gives up when signing contract c; and
- A *penalty* $P_{c,X}$ that agent X receives if the contract fails because of the other party.

Such values are not restricted to be monetary quantifications, rather they quantify of the level of satisfaction of X. All in all, such quantities are subjective and therefore we cannot assess any mathematical relations between values of the triples of two different agents, even though they refer to the same contract.

More in details, a contract c has the following properties from the point of view of X:

- If the contract is honoured, agent X will receive $R_{c,X}$ with probability one; and
- If the contract fails because of agent Y, agent X will receive $P_{c,X}$ with probability one.

Another assumption concerns the relative ordering of reward, investment and penalty in a single subjective evaluation. We are interested in contracts whose parameters are ordered as follows:

$$P_{c,X} \leq I_{c,X} \leq R_{c,X}$$

This inequality captures the essence of risky contracts. Moreover, it implies that we are interested in agents that sign contracts with the intent of honouring them. Any failure in honouring a contract turns into a loss of utility (see later on): $I_{c,X} - P_{c,X}$. Furthermore, agents in our model do not consider their failure in honouring a contract, they assume that they can honour all contracts they sign; nevertheless the uncertainty about the other signer remains.

Guarantor. The abstraction of Guarantors was introduced and discussed in details in [1,2]. For the sake of completeness, we can simply say that here Guarantors are sources of highly trusted information and they are trust catalysts. If agent X requests a piece of information to Guarantor G, it assigns a correctness probability of one to the received response. Nevertheless, we introduce some failure probability in order to account for the idea that the use of additional information, i.e., the information that Guarantor provides, always introduces some risk, even though the information source is highly trusted and reliable.

2.2 Expectation of the Utility of Agents

The rest of this section analyses the utility that agents estimate in the process of signing a contract. This utility is formalised with and without the mediation of a Guarantor, and then such two cases are compared.

Direct interaction. Taking into account the previous definitions, we can explicitly write the expected value of the utility that agent X receives from a contract with agent Y:

$$\overline{U}^r_{X,c} = R_{c,X} \cdot p_{c,Y} + P_{c,X} \cdot (1 - p_{c,Y})$$

where the superscript "r" indicates that the real probability is used in this equation, and not an estimation of its value. This utility is not available to any agent since $p_{c,Y}$ is not observable. Instead, agent X estimates the expected utility using its trust in the other party (agent Y):

$$\overline{U}^e_{X,c} = R_{c,X} \cdot t_{c,X,Y} + P_{c,X} \cdot (1 - t_{c,X,Y})$$

Taking into account that agent X invests a certain value when it signs the contract, and that any contract has some probability $p^s_{c,X}$ of being finally signed, the total average utility that agent X perceives is:

$$\begin{aligned}\overline{U}^r_X &= \overline{U}^r_{X,c} \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X}) = \\ &= [R_{c,X} \cdot p_{c,Y} + P_{c,X} \cdot (1 - p_{c,Y})] \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X})\end{aligned}$$

As before, the agent can only estimate the total utility, obtaining:

$$\begin{aligned}\overline{U}^e_X &= \overline{U}^e_{X,c} \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X}) = \\ &= [R_{c,X} \cdot t_{c,X,Y} + P_{c,X} \cdot (1 - t_{c,X,Y})] \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X})\end{aligned}$$

Guarantor-mediated interaction. We can adapt the previous equations to the case in which the contract is evaluated using additional information obtained from a Guarantor.

In this case, the failure probability that we associate with a Guarantor has to be considered. This failure probability accounts for the possible uncertainty of the information that the Guarantor provides.

In particular, we assume that an error of a Guarantor may cause a failure of the contract. In this case agent X receives $P_{c,X}$. This risk is acceptable if we assume that in the case of an error, the Guarantor itself, and not contractors, pays the penalty.

Under this assumption, the new expected value of the utility of signing contract c is:

$$\overline{U}^{G,r}_{X,c} = R_{c,X} \cdot P\{c \text{ honoured}\} + P_{c,X} \cdot P\{c \text{ not honoured}\}$$

where the G superscript indicates that some information from the Guarantor is considered when signing the contract.

Under the assumption that p^G_k is the probability of the Guarantor to provide erroneous information and that any error of the Guarantor immediately causes the contract to fail, it is possible to express the total contract success and failure probabilities:

$$P\{c \text{ honoured}\} = p_{c,Y} \cdot p^G_k \tag{1}$$

$$\begin{aligned} P\{c \text{ not honoured}\} &= P\{c \text{ not honoured}|\text{Guarantor succeeds}\} + \\ &+ P\{c \text{ not honoured}|\text{Guarantor fails}\} \\ &= (1 - p_{c,Y}) \cdot p^G_k + 1 - p^G_k = 1 - p_{c,Y} \cdot p^G_k \end{aligned} \tag{2}$$

This allows rewriting the previous Equation (1) as:

$$\overline{U}^{G,r}_{X,c} = R_{c,X} \cdot p_{c,Y} \cdot p^G_k + P_{c,X} \cdot (1 - p_{c,Y} \cdot p^G_k)$$

Then, exploiting this equality in (1) we obtain the total average utility of signing the contract using information from a Guarantor as:

$$\begin{aligned} \overline{U}^{G,r}_{X} &= \overline{U}^{G,r}_{X,c} \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X}) = \\ &= [R_{c,X} \cdot p_{c,Y} \cdot p^G_k + P_{c,X} \cdot (1 - p_{c,Y} \cdot p^G_k)] \cdot p^s_{c,X} + \\ &\quad + I_{c,X} \cdot (1 - p^s_{c,X}) \end{aligned}$$

Since agents give a trust of one to their Guarantors, most of the estimations of agent X are not changed by the mediation. In particular, the estimation of the contract success probability remains unchanged; therefore the estimation of the average utility of the contract does not change. Also, the estimation of the expected utility as a function of the probability of signing (1) is not influenced. As explained later on, the mediation of the Guarantor influences only the decision making strategy.

3 Decision Making Strategy

In this section, we introduce a rationality principle in our model by means of a decision making strategy that exploits utility to discriminate on the inclusion of the mediation of a Guarantor into an interaction.

3.1 Probability Density Function of Trust and the Risk Factor

As we said in Section 2, we model trust from the point of view of an agent as the estimation of the probability of having a contract honoured by its counterpart. An underlying assumption of this definition is that this estimation, and the real probability of the contract being honoured, both lie in the interval $[t_{min}, t_{max}]$. In essence, trust is a random variable $\mathbf{t}$ whose Probability Density Function (PDF) depends on decision making strategies of the agents involved in the contract.

Taking the variable $\mathbf{t}$ and a rationality principle into account, it is easy to define the probability that agent X would sign a given contract c.

In particular, this reasonable rationality principle mentioned above states that:

X decides to sign a contract c with Y if the estimated expected utility that it perceives is greater than the investment required to sign the contract

Which, in formal terms, is:

$$\begin{aligned} p^s_{c,X} &\doteq P\{\overline{U}^e_{X,c} > I_{c,X}\} = \\ &= P\{R_{c,X} \cdot t_{c,X,Y} + P_{c,X} \cdot (1 - t_{c,X,Y}) > I_{c,X}\} \end{aligned}$$

A further simple elaboration of this equation yields:

$$\begin{aligned} p^s_{c,X} &= P\{t_{c,X,Y} \cdot (R_{c,X} - P_{c,X}) > I_{c,X} - P_{c,X}\} = \\ &= P\left\{t_{c,X,Y} > \frac{I_{c,X} - P_{c,X}}{R_{c,X} - P_{c,X}}\right\} \end{aligned}$$

Where we supposed that $R_{c,X} - P_{c,X}$ is not zero. Now, if we define:

$$\kappa_{c,X} \doteq \frac{I_{c,X} - P_{c,X}}{R_{c,X} - P_{c,X}}$$

it is possible to express $p^s_{c,X}$ as:

$$p^s_{c,X} = P\{t_{c,X,Y} > \kappa_{c,X}\} \tag{3}$$

This last equation indicates that agent X signs contract c if its trust in the counterpart with respect to c exceeds $\kappa_{c,X}$, that we call *risk factor*. This factor depends only on X's subjective evaluation of contract c and it describes the risk that X perceives in signing contract c. This, allows to rephrase the decision making strategy as:

Agent X signs a contract c with a counterpart Y if and only if its trust in Y for contract c is greater then the risk factor of c.

It is worth noting that risk factor $\kappa_{c,X}$ is a number between zero and one. Furthermore, it is the quotient of two quantities that have a precise meaning on their own:

- The numerator $N_{c,X} = I_{c,X} - P_{c,X}$ expresses the gain that agent X obtains when rejecting contract c, in comparison to the case in which the contract is accepted but actually not honoured.
- The denominator $H_{c,X} = R_{c,X} - P_{c,X}$ represents the gain that the contract yields in case of success with respect to failure.

Then, e.g., if we consider the boundary cases:

- $\kappa_{c,X} = 1$ means that the contract will never be signed, because the investment equals the utility, but the first is guaranteed while the second is not.
- $\kappa_{c,X} = 0$ means that the contract has no risk, since the investment equals the penalty (which is assured with probability one). Therefore the contract will always be accepted.

In particular, if $\kappa_{c,X} \leq t_{min}$ the contract is always rejected, while if $t_{max} \leq \kappa_{c,X}$ the contract is always accepted. This consideration accounts also for the boundary cases analysis exposed above.

Having introduced the risk factor $\kappa_{c,X}$, it is possible to rewrite Equation (1) putting some emphasis on it. In particular:

$$\begin{aligned}\overline{U}^r_X &= [R_{c,X} \cdot p_{c,Y} + P_{c,X} \cdot (1 - p_{c,Y})] \cdot p^s_{c,X} + I_{c,X} \cdot (1 - p^s_{c,X}) = \\ &= [(R_{c,X} - P_{c,X})p_{c,Y} + P_{c,X}] \cdot p^s_{c,X} + I_{c,X}(1 - p^s_{c,X}).\end{aligned}$$

Now, explicitly showing $p^s_{c,X}$ and subsequently $(R_{c,X} - P_{c,X})$:

$$\begin{aligned}\overline{U}^r_X &= [(R_{c,X} - P_{c,X})p_{c,Y} + P_{c,X} - I_{c,X}] \cdot p^s_{c,X} + I_{c,X} = \\ &= (R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X}) \cdot p^s_{c,X} + I_{c,X}.\end{aligned}$$

This last equation gives the possibility to draw some interesting considerations. First, $\overline{U}^r_X$ is bounded between $P_{c,X}$ and $R_{c,X}$. Furthermore, $\overline{U}^r_X$ is a linear function of $p^s_{c,X}$, and its slope is $(R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X})$. Since $(R_{c,X} - P_{c,X})$ is non negative because of (2.1), the sign of the slope is influenced by $(p_{c,Y} - \kappa_{c,X})$ only. This ultimately means that the risk factor is an indicator of convenience in terms of average utility:

- If the success probability of the contract is greater than $\kappa_{c,X}$, then the average utility (of X) increases with the probability of signing the contract, i.e., the contract is advantageous.
- If the risk factor is lower than $\kappa_{c,X}$, the contract is disadvantageous and the average utility decreases with $p^s_{c,X}$.
- If $\kappa_{c,X} \equiv p_{c,Y}$, the average utility is constant.

3.2 Role of the PDF of Trust

The only working assumption that we took up to now is that $\mathbf{t}$ is a random variable bound by t_{min} and t_{max}. In this section, we further elaborate on trust as a random variable and, without breaking our blackbox approach, we go for the worst case and we assume that $\mathbf{t}$ is uniformly distributed in interval $[t_{min}, t_{max}]$. This new assumption allows us to study the influence of the mediation of a Guarantor on the average utility perceived by agents.

In accordance with Equation (3), we can express the signing probability as the probability that $t_{c,X,Y} \geq \kappa_{c,X}$. Therefore:

$$p^s_{c,X} = P\{t_{c,X,Y} > \kappa_{c,X}\} = \int_{\kappa_{c,X}}^{+\infty} f(t_{c,X,Y})\, \mathrm{d}t_{c,X,Y} \tag{4}$$

Then,

$$p^s_{c,X} = \begin{cases} 1 & \kappa_{c,X} \leq t_{min} \\ \frac{t_{max} - \kappa_{c,X}}{t_{max} - t_{min}} & t_{min} < \kappa_{c,X} < t_{max} \\ 0 & t_{max} \leq \kappa_{c,X} \end{cases} \tag{5}$$

Figure 1 shows a pictorial description of this last equation.

Now, we focus our analysis of the utility on the case in which $t_{min} \leq \kappa_{c,X} \leq t_{max}$, i.e., we exclude the edge cases. Moreover, we assume symmetric PDF of $\mathbf{t}$.

Introducing (5) in (4) we obtain the average utility as a function of t_{min} and t_{max}:

$$\overline{U}^r_X = (R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X}) \cdot \frac{t_{max} - \kappa_{c,X}}{t_{max} - t_{min}} + I_{c,X}. \tag{6}$$

Then, using a symmetric PDF of $\mathbf{t}$ with width δ it is possible to rewrite (5) as:

$$p^s_{c,X} = \begin{cases} 1 & \kappa_{c,X} \leq t_{min} \\ \frac{t_{max} - \kappa_{c,X}}{t_{max} - t_{min}} & t_{min} < \kappa_{c,X} < t_{max} \\ 0 & t_{max} \leq k \end{cases} \tag{7}$$

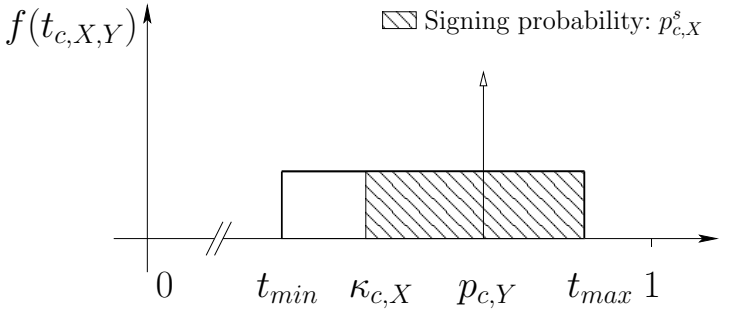


Fig. 1. The process of estimation of trust and its influence in decision making

And then:

$$p^s_{c,X} = \begin{cases} 1 & \kappa_{c,X} \leq t_{min} \\ \frac{p_{c,Y}+\delta-\kappa_{c,X}}{2\delta} & t_{min} < \kappa_{c,X} < t_{max} \\ 0 & t_{max} \leq k \end{cases}$$

that expresses $p^s_{c,X}$ as a function of δ. Substituting this equation in Equation (6) and excluding the edge cases yields to:

$$\overline{U}^r_X = (R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X}) \cdot \frac{p_{c,Y} + \delta - \kappa_{c,X}}{2\delta} + I_{c,X}. \tag{8}$$

This equation expresses the average utility as a function of the width of the probability density function of $\mathbf{t}$. Since the utility is a hyperbolic function of δ, any small decrease of δ implies a much higher increase in the average utility and vice versa. This introduces one of the main considerations of the paper:

If the information provided by a Guarantor linearly narrows the width of the PDF of $\mathbf{t}$*, then a hyperbolic increase of the average utility of having the contract signed occurs.*

Regarding the edge cases, when $\kappa_{c,X} \leq t_{min}$ or $t_{max} \leq \kappa_{c,X}$, the probabilities of signing the contract are one and zero respectively (5), and the utility is constant with respect to δ. Figure 2 shows the behaviour of the average utility for an increasing value of δ. The first plateau expresses the case in which $\kappa_{c,X} \leq t_{min}$, and consequently $p^s_{c,X}$ equals one.

Equation (8) has the following interesting consequence on the behaviour of the utility. If agent X takes its decisions on signing a contract c using a symmetric

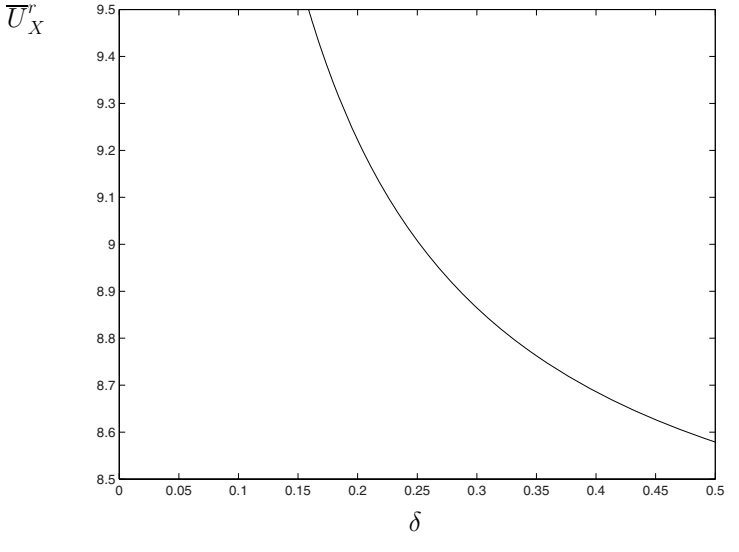


Fig. 2. Hyperbolic decrease of $\overline{U}^r_X(\delta)$

PDF for $\mathbf{t}$ centred in $p_{c,Y}$ and if the contract does not fail because of X, then $\forall \delta \in \mathbf{R} : \delta \geq 0,\ t_{min} - \delta \geq 0,\ t_{max} + \delta \leq 1, \overline{U}^r_X(\delta)$ is non-increasing. In fact, $\overline{U}^r_X$ is piecewise differentiable and the differentiation of (8) for $t_{min} < \kappa_{c,X} < t_{max}$ yields to:

$$\frac{\partial \overline{U}^r_X}{\partial \delta} = (R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X}) \cdot \frac{2\delta - 2(p_{c,Y} - \kappa_{c,X} + \delta)}{4\delta^2} = \\ = -\frac{(R_{c,X} - P_{c,X}) \cdot (p_{c,Y} - \kappa_{c,X})^2}{2\delta^2}$$

Taking into account that a subjective evaluation is well formed if $R_{c,X} \geq P_{c,X}$, the partial derivative is always non-positive, i.e., any enlargement of the estimation (which introduces uncertainty), worsen the performance of the agent's decision strategy and its relative utility.

The explicit choice of a PDF for trust $\mathbf{t}$ allows elaborating on the inclusion of mediation into an interaction. The two parameters $\kappa_{c,X}$ and $p_{c,Y}$ are kept fixed, since the mediation of a Guarantor does not change or influence them. On the contrary, the total error probability is modified to account for the additional probability of error that the Guarantor brings. Using Equation (1), it is possible to directly substitute $p_{c,Y}$ with $p_{c,Y}\, p^G_k$ to express the total success and failure probabilities, thus obtaining the equivalent of (8) for the case of Guarantor-mediated interactions. To stress the fact that the width of the estimation is different when introducing a Guarantor in the interaction, we use δ^G instead of δ:

$$\overline{U}^{G,r}_X = \begin{cases} H_{c,X} \cdot M^G + I_{c,X} & \kappa_{c,X} \leq t_{min} \\ H_{c,X} \cdot M^G \cdot \frac{p_{c,Y} + \delta^G - \kappa_{c,X}}{2\delta^G} + I_{c,X} & t_{min} < \kappa_{c,X} < t_{max} \\ I_{c,X} & t_{max} \leq \kappa_{c,X} \end{cases} \tag{9}$$

Where we defined (see next section) $M^G = (p_{c,Y}\, p^G_k - \kappa_{c,X})$.

However, it is important to note that this estimation is always centred around $p_{c,Y}$, since agent X accords a trust of one to its Guarantor.

4 Results and Bounds

In this section we study the effects of mediation in our model. In order to do so, we recall that our working assumption is that Guarantors provide additional information to agents, thus allowing for a more precise (narrower) estimation of probability $p_{c,Y}$. Anyway, Guarantors, although highly reliable, introduce additional error probability, that must be compensated by improvements in the estimation of trust.

In order to quantify the performance of a Guarantor as a middleman in an interaction between agent X and Y, we calculate the amount of additional information that a Guarantor needs to provide in order to keep the average utility of agent X fixed.

The comparison of the two utilities expressed in Equations (8) and (9) allows to calculate the width of Guarantor-mediated estimation of trust for which the

utility equals the case without mediation. If we introduce $\hat{\delta}^G$, a function of δ and p_k^G, as:

$$\hat{\delta}^G \doteq \delta_G \in \left[0, \frac{1}{2}\right] : \overline{U}^r_X(\,\delta\,,\, p_{c,Y}\,) = \overline{U}^{G,r}_X(\,\delta_G\,,\, p_{c,Y} \cdot p_k^G\,)$$

we can compare Equations (8) and (9) to obtain:

$$(p_{c,Y} - \kappa_{c,X}) \cdot \frac{p_{c,Y} + \delta - \kappa_{c,X}}{\delta} = (p_{c,Y}\, p_k^G - \kappa_{c,X}) \cdot \frac{p_{c,Y} + \hat{\delta}^G - \kappa_{c,X}}{\hat{\delta}^G}$$

where we subtracted $I_{c,X}$ on both sides and multiplied by $\frac{2}{R_{c,X} - P_{c,X}}$. Then, introducing $M = (p_{c,Y} - \kappa_{c,X})$ and $M^G = (p_{c,Y}\, p_k^G - \kappa_{c,X})$:

$$M \cdot \frac{\delta + M}{\delta} = M^G \cdot \frac{\hat{\delta}^G + M}{\hat{\delta}^G}$$

and dividing by M^G yields:

$$\frac{M}{M^G} \cdot \frac{\delta + M}{\delta} = \frac{\hat{\delta}^G + M}{\hat{\delta}^G}.$$

This last equation allows to make $\hat{\delta}^G$ explicit:

$$\hat{\delta}^G = \frac{M}{\frac{M}{M^G} \cdot \frac{\delta + M}{\delta} - 1} = \frac{M\, M^G\, \delta}{M(\delta + M) M^G\, \delta}$$

that holds if $M^G \neq 0$.

It should be quite clear that $\hat{\delta}^G$ is the breakeven point that makes agent X choose to go for a mediated interaction rather than for a direct interaction:

- If the Guarantor provides enough information to restrict the estimation of trust to a width less than $2\,\hat{\delta}^G$, the use of the mediation is advantageous.
- If the estimation remains larger than $2\,\hat{\delta}^G$, the error probability introduced by the Guarantor decreases the average utility.

It is worth noting that this decision strategy is purely ideal because agent X does not know p_k^G. Anyway, Figure 3 provides an ideal means for evaluating the zone of convenience for choosing mediated interactions.

In order to ground our model to everyday experience, we recall that we are interested in Guarantors that introduce a very low probability of error, and therefore we study the behaviour of $\hat{\delta}^G$ as p_k^G tends to one. What we obtain from this study is that if agent X makes its decisions using a symmetric PDF and that the contract does not fail because of X, $\forall \delta \in \mathbf{R} : 0 \leq \delta \leq \frac{1}{2}$, $\delta - \hat{\delta}^G$ tends to zero in an hyperbolic way as p_k^G tends to one. Due to the lack of space

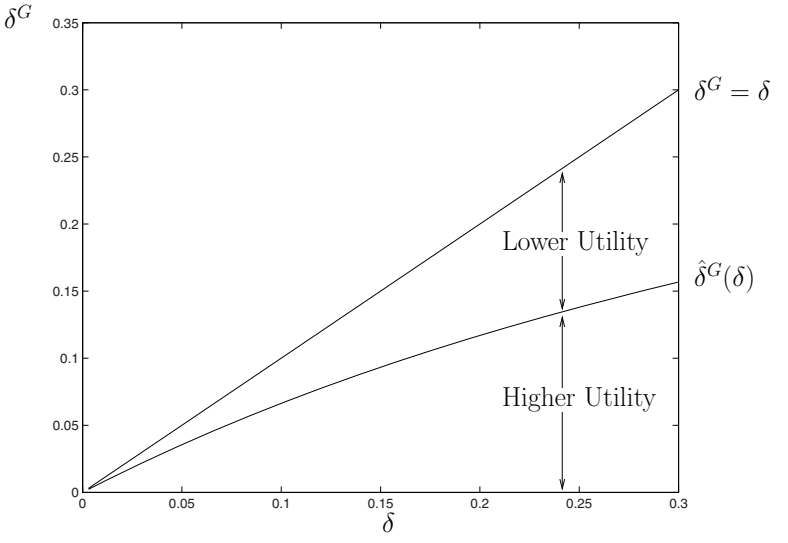


Fig. 3. Zone of convenience for mediated interactions

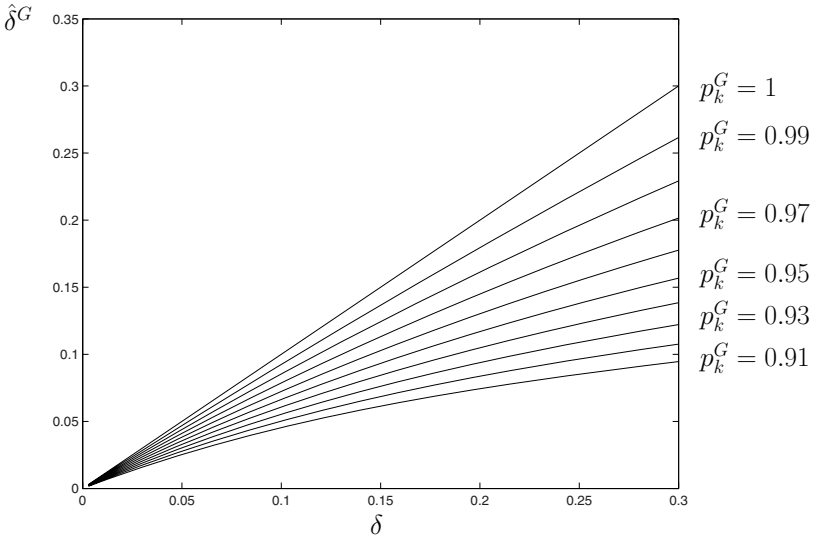


Fig. 4. $\hat{\delta}^G$ as a function of δ for different values of p_k^G

in this paper, we cannot go in the details of the demonstration of this result, anyway it is worth saying that the demonstration is divided into four steps:

- Proof that $\hat{\delta}^G$ is a hyperbolic function of p_k^G;
- Proof that $\hat{\delta}^G$ is increasing from a certain value of p_k^G on, excluding the edge cases;

– Proof that its maximum value is δ;
– Proof for the edge cases.

This result shows that if a Guarantor introduces a (sufficiently) low probability of error, the use of its mediation is advantageous and the advantages that it brings are fast increasing as the probability of error decreases. As a further evidence of the convenience of using Guarantor-mediated interaction, the behaviour of the zone of convenience with respect to p_k^G is shown in Figure 4.

5 Conclusions

The aim of this work is to provide a sound demonstration that the development of Guarantor-mediated infrastructures is extremely beneficial to support secure, trusted and privacy-aware interactions in real-world societies of agents. In particular, such infrastructures provide notable features that are not discussed here, but that play a fundamental role from the point of view of scalability, reliability and traceability (see [1,2]). Then, in many cases, the additional utility that mediation provides to agents is considerable even through Guarantors are not error-free.

This work is not meant to be conclusive and many points remain open. One of the major planned developments regards the study of concrete trust estimators, and the introduction of the resulting PDFs in our model. Another very important open point regards the study of the effects of delegation of tasks and goals through a chain of delegated Guarantors.

Furthermore, the study of one of the main features of Guarantors, i.e., the possibility of anonymising interactions, is still in search of a formalization (and of a probabilistic model), even though its characteristics and possible uses are clearly understood [1]. This kind of interaction allows to prevent unwanted spread of sensible information; as such, its study remains central in the evaluation of the agent's benefits from the Guarantor infrastructure.

Acknowledgements

This work is partially supported by project CASCOM (FP6-2003-IST-2/511632). The CASCOM consortium is formed by DFKI (Germany), TeliaSonera AB (Sweden), EPFL (Switzerland), ADETTI (Portugal), URJC (Spain), EMA (Finland), UMIT (Austria), and FRAMeTech (Italy). The authors would like to thank all partners for their contributions.

References

1. Bergenti, F., Bianchi, R., Fontana, A.: Secure and Trusted Interactions in Societies of Electronic Agents. In: Proceedings of The 4th Workshop on the Law and Electronic Agents (LEA 2005), Bologna, Italy, pp. 1–12. Wolf Legal Publishers (2005)

2. Bianchi, R., Fontana, A., Bergenti, F.: A Real–World Approach to Secure and Trusted Negotiation in MASs. In: Proceedings of the 4th International Joint Conference on Agents and Multi-Agents Systems (AAMAS), pp. 1163–1164. ACM Press, New York (2005)
3. Bons, R.W.H.: Designing Trustworthy Trade Procedures for Open Electronic Commerce. Ph.D.diss., EURIDIS and Faculty of Business Administration, Erasmus University, Rotterdam, The Netherlands (1997)
4. CASCOM Web site, http://www.ist-cascom.org
5. Castelfranchi, C., Falcone, R.: Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification. In: Proceedings of the International Conference of Multi-agent Systems (ICMAS), pp. 72–79. ACM Press, New York (1998)
6. Debenham, J., Sierra, C.: An Information-based Model for Trust. In: Proceedings of the 5th International Conference on Autonomous Agents and Multiagent systems (AAMAS), Utrecht, The Netherlands, pp. 497–504 (2005)
7. Ellison, C.: SPKI Requirements. IETF RFC 2692 (1999)
8. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. IETF RFC 2693 (1999)
9. Gambetta, D. (ed.): Trust: Making and Breaking Co-operative Relations. Basil Blackwell, Inc., Malden (1988)
10. JENA Web site, http://jena.sourceforge.net
11. Jennings, N.R., Parsons, S., Sierra, C., Faratin, P.: Automated Negotiation. In: Proceedings of the 5thInternational Conference on the Practical Application of Intelligent Agents and Multi-Agents Systems (PAAM 2000), Manchester, UK, pp. 23–30 (2000)
12. Jøsang, A., Ismail, R., Boyd, C.: A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems 43(2), 618–644 (2007)
13. Marsh, S.: Formalising Trust as a Computational Concept. Ph.D. diss., Department of Mathematics and Computer Science, University of Stirling, Stirling, UK (1994)
14. MINDSWAP Team. A Definition of Trust for Computing with Social Networks Technical report, University of Maryland, College Park (February 2005)
15. OWL Web site, http://www.w3.org/2004/OWL
16. Racer Web site, http://www.racer-systems.com
17. Yu, B., Singh, M.: Searching Social Network. In: Proceedings of The 2nd International Joint Conference On Autonomous Agents and Multiagent Systems. ACM Press, New York (2003)

Arguing about Reputation: The LRep Language

Isaac Pinyol and Jordi Sabater-Mir

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Scientific Research Council
Bellaterra, Barcelona, Spain
{ipinyol,jsabater}@iiia.csic.es

Abstract. Since electronic and open environments became a reality, computational models of trust and reputation have attracted increasing interest in the field of multi-agent systems (MAS). In virtual societies of human actors very well-known mechanisms are already used to control non normative agents, for instance, the eBay scoring system. In virtual societies of artificial and autonomous agents, the same necessity arises, and several computational trust and reputation models have appeared in literature to cover this necessity. Typically, these models provide evaluations of agents' performance in a specific context, taking into account direct experiences and third party information. This last source of information is the communication of agents' own opinions. When dealing with cognitive agents endowed with complex reasoning mechanisms, we would like that these opinions could be justified in a way such that the resulting information was more complete and reliable. In this paper we present LRep, a language based on an existing ontology of reputation that allows building justifications of communicated social evaluations.

1 Introduction

The field of multiagent systems has experienced an important growth and evolution in the past few years. These systems can be seen as virtual societies composed of autonomous agents where there is a need to interact with other members of the society to achieve their goals. As in human societies, these interactions usually involve an exchange of information. The problem of partners selection via the detection of good or bad potential partners, or how agents evaluate the credibility of received information, arises in a scenario like this. Human societies, throughout history, have been using trust and reputation mechanisms for this purpose. These powerful social control artifacts have been studied from different perspectives, such as psychology (Bromley [1], Karlins et al. [2]), sociology (Buskens [3]), philosophy (Plato [4], Hume [5]) and economics (Marimon et al. [6], Celentani et al. [7]).

In multiagent systems the interest in these mechanisms has considerably increased and, as a consequence, numerous computational trust and reputation models have appeared in the literature. E-Commerce sites already use some of them (eBay [8], Amazon [9], OnSale [10]). These models consider reputation as a centralized global property. So, the reputation of each agent is public and all

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 284–299, 2008.

agents perceive the same reputation value. More sophisticated models ([11], [12], [13], [14], [15], [16], [17]) consider reputation as a subjective property. Therefore every agent has its own reputation system that provides evaluations of other agents calculated from external communication and direct experience, giving the agent its own vision of the society. Furthermore, other models (see [18], [19]) take into account social information when providing these evaluations.

One of these models is Repage [17], a computational system based on a cognitive theory of reputation. This model is designed to be part of a cognitive agent,i.e., an agent endowed with beliefs, desires and intentions. Like other reputation models, Repage uses social evaluations obtained from direct experiences and communicated social evaluations as source for calculations. However, this communication is quite simple and very limited, allowing only the exchange of single values associated with a reliability measure. In a real environment and for an agent that is able to make complex reasoning, an opinion without being justified can be very weak and not as useful as a fully justified opinion that points out where the information is coming from. With agents using a complex reputation model like Repage, it can be as important to know the followed procedure and the sources used to calculate the final value, as the final value itself.

In this paper we present LRep, a simple language that can be used with a model like Repage to elaborate justifications of calculated values. These justifications can have different levels of detail. So, agents can decide the amount of extra information and the level of detail of them when there are communicating social evaluations.

In Section 2 we briefly introduce Repage and its theory framework. Following this, in Section 3 we introduce an ontology of reputation and its specification using description logics. This ontology will be used to define the semantics of LRep. Afterwards, in Section 4 we define the syntax and semantics of LRep. In Section 5 we present several situations where the use of LRep and justification helps to improve the performance of cognitive agents. Finally, Section 6 presents the conclusions and future work.

2 The Repage System

In order to present the Repage system it is necessary to get in touch with the theoretical framework upon which it is based. This framework is a cognitive theory of reputation developed by Conte and Paolucci in [20]. In this book they study the impact of the transmission of social evaluations in artificial societies, pointing out the important difference between information that is thought to be true and information that is said.

This theory describes a model of imAGE, REPutation and their interplay. Although both are social evaluations, image and reputation are distinct objects. Image is a simple evaluative belief; it tells that the target is “good” or “bad” with respect to a norm, a standard, or a skill. Reputation is a belief about the existence of a communicated evaluation. Consequently, to assume that a target t is assigned a given reputation implies only to assume that t is reputed to be

"good" or "bad", i.e., that this evaluation circulates, but it does not imply to share the evaluation.

To select good partners, agents need to form and update own social evaluations; hence, they must exchange evaluations with one another. If agents should transmit only believed image, the circulation of social knowledge would be bound to stop soon. On the other side, agents that believe all the informations that they receive would be no more autonomous; in order to preserve their autonomy, agents need to *decide* independently whether to share or not and whether to believe or not others' evaluations of a given target. Hence, they must:

- Form both evaluations (image) and meta-evaluations (reputation), keeping distinct the representation of own and others' evaluations, before
- Deciding whether or not to integrate reputation with their own image of a target.

Unlike other current systems, in Repage reputation does not coincide with image. Indeed, agents can either transmit their own image of a given target, which they hold to be true, or report on what they have "heard" about the target, i.e. its reputation, whether they believe this to be true or not. Of course, in the latter case, they will neither commit to the information truth value nor feel responsible for its consequences. Consequently, agents are expected to transmit uncertain information, and a given positive or negative reputation may circulate over a population of agents even if its content is not actually believed by the majority.

2.1 The Repage Architecture

The Repage architecture (see figure 1) was designed to reflect the distinction between image and reputation. It has three main elements: a memory, a set of detectors and the analyzer. The memory is composed of a set of inter-connected predicates that are conceptually organized in different levels of abstraction. Each predicate that belongs to one of the main types, the ones showed in figure 1, contains a probabilistic evaluation that refers to a certain target agent in a specific role. For instance, an agent may have an image of agent T (target) as a seller (role), and a different image of the same agent T as informant. The evaluation consist of a probability distribution over the discrete sorted set of labels: {Very Bad, Bad, Normal, Good, Very Good}.

The network of dependences specifies which predicates contribute to the values of others. In this sense, each predicate has a set of antecedents and a set of consequents. The detectors, inference units specialized in each particular kind of predicate, receive notifications from predicates that changes or that appear in the system and uses the dependences to recalculate the new values or to populate the memory with new predicates.

Each predicate has associated a strength that is a function of its antecedents and of the intrinsic properties of each kind of predicate. As a general rule, predicates that resume or aggregate a bigger number of predicates will hold a higher strength.

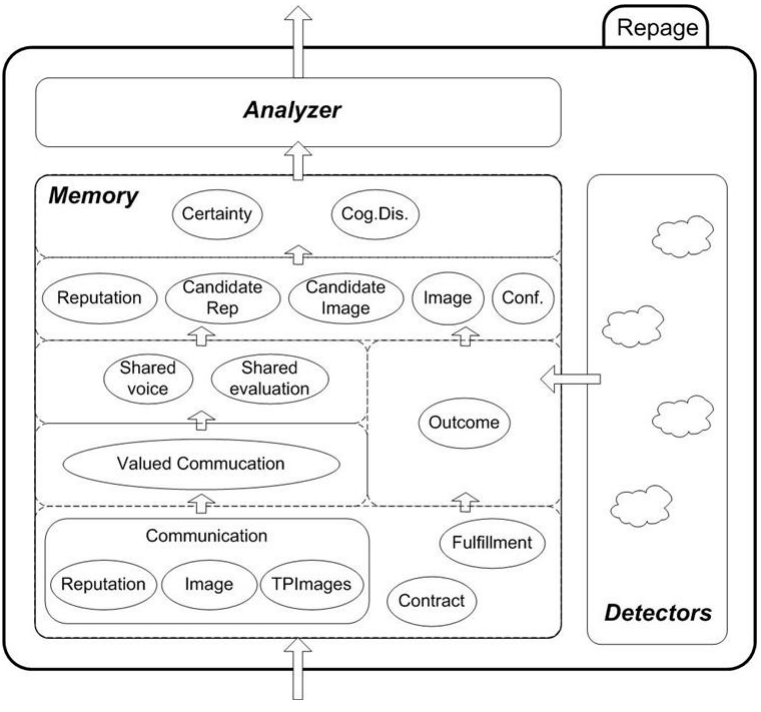


Fig. 1. The Repage architecture

At the first level of the Repage memory we find a set of predicates not evaluated yet by the system. *Contracts* are agreements on the future interaction between two agents. Their result is represented by a *Fulfillment*. *Communications* is information that other agents may convey, and may be related to three different aspects: the image that the informant has about a target, the image that, according to the informant, a third party agent has on the target, and the reputation that the informant has about the target.

In level two we have two kind of predicates. *Valued communication* is the subjective evaluation of the communication received that takes into account, for instance, the image the agent may have of the informant as informant. Communications from agents whose credibility is low will not be considered as strong as the ones coming from well reputed informants. An *outcome* is the agent's subjective evaluation of a direct interaction, built up from a fulfillment and a contract.

At the third level we find two predicates that are only fed by valued communications. On the one hand, a *shared voice* will hold the information received about the same target and same role coming from communicated reputations. On the other hand, *shared evaluation* is the equivalent for communicated images and third party images.

Shared voice predicates will finally generate *candidate reputation*; shared evaluation together with outcomes will generate *candidate image*. Newly generated candidate reputation and image aren't usually strong enough; new communications and new direct interactions will contribute to reinforce them until a threshold, over which they become full-fledged image or reputation. We refer to [17] for a much more detailed presentation.

From the point of view of the agent structure, integration with the other parts of our deliberative agents is straightforward. Repage memory links to the main memory of the agent that is fed by its communication and decision making module, and at the same time, this last module, the one that contain all the reasoning procedures uses the predicates generated by Repage to make decisions.

3 The Ontological Dimension of Reputation

As we have shown so far, reputation mechanisms play a crucial role in the way we conceive agents' societies. But social evaluations are more than simple ratio scores. In cognitive agents the fact of acknowledging certain reputation or image of other agents imply a mental state, a set of beliefs about the future performance of target agents, but at the same time, the formation of such a high level predicates, require several intermediate cognitive steps, that generate a full taxonomy of interrelated predicates. From this point of view, it is easy to think about an ontology of reputation and image showing this structure. A possible ontology is defined in [21].

The concepts that appear in this ontology are very similar to the typology of predicates that Repage defines. In [21] we define a mapping between Repage predicates and the ontology (that is almost direct). Still though, we want to use as source of information this common ontology, since is not linked to any

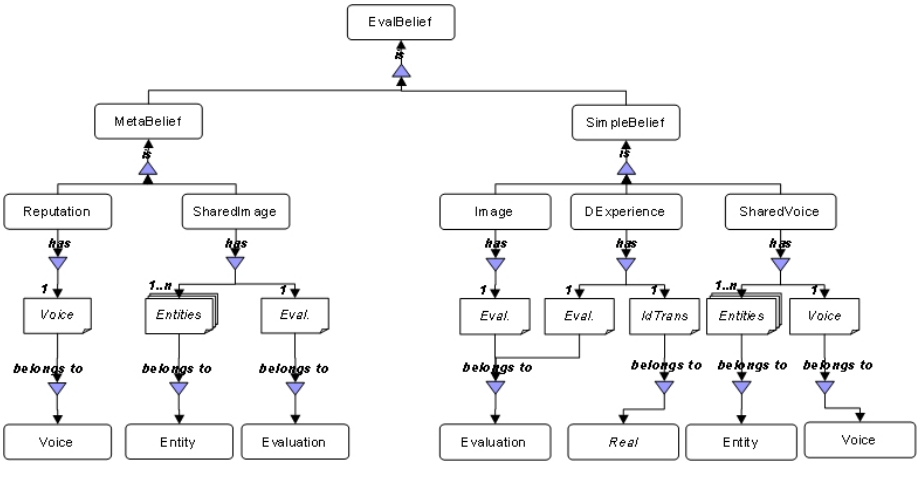


Fig. 2. The taxonomy, membership relations and components of evaluative beliefs

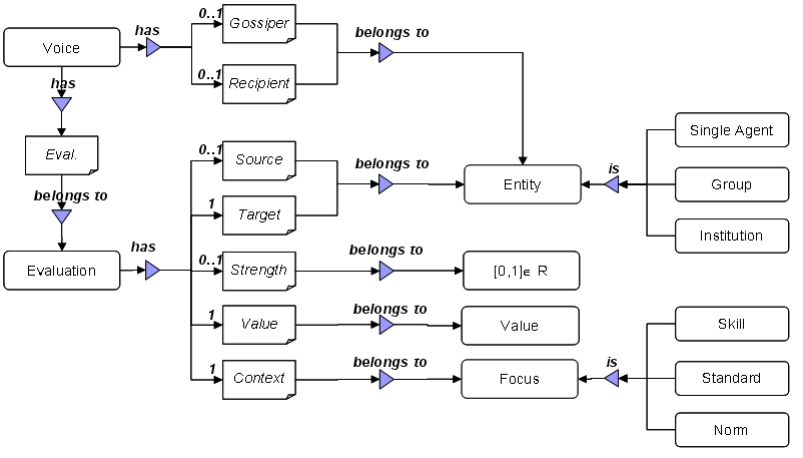


Fig. 3. The main classes and components of a social evaluation and voice

particular reputation model. A graphical representation of it is shown in Figure 2 and 3. Nevertheless, we need a more formal approach. Because LRep language is based on this ontology we need a formalism that allows us to refer instances of its concepts. For that, we decided to use description logics (DL). As we will explain, DL offers an elegant way to represent application domains, and its concepts have been used for the semantic web (in term of the language OWL DL) to describe ontologies. Furthermore, its syntax and semantics are very well defined (see [22]). In this section we first make a short introduction to what is a description logic system and why it is a good option to use as an ontology formalism. Afterwards, we give a description of the ontology using this formalism.

3.1 Description Logic

Description Logic (DL) is a knowledge representation formalism used to represent the application domain, the world. Its power relies on the formal logic-based semantics and the reasoning engine with which it is equipped. A DL system has two differentiate submodules, *TBox* and *ABox*.

On the one hand, the *TBox* contains a set of expressions in one of the languages of the $\mathcal{AL}$-languages family (see [22]), that define the terminology of the domain (the classes). This family of languages can be seen as fragments of first-order Logic(FOL) [22][1], but its expressiveness simplifies the formulas and is specially suited for the definition of concepts. On the other hand, the *ABox* contains assertions about named individuals in terms of the terminology defined in the *TBox*, the state of the world. In general, a knowledge representation system based on DL provides facilities to set and update knowledge bases, to manipulate it and to reason over it.

[1] So, all formulas of $\mathcal{AL}$-languages can be expressed as FOL formulas keeping the same semantic.

Because DL systems use segments of FOL, the set of predicates contain implicit knowledge, that can be made explicit using inference. Thus, the concept of satisfiability is defined in the classical way (see [22]). Having a DL system D, a concept C and an individual element a , we say that $D \models C(a)$ iff $C(a)$ can be inferred from D. In this case this is deducible using some of the reason algorithms defined for DL systems (Structural Subsumption Algorithm or Tableau Algorithm, for instance)[22].

Nowadays, the interest in DL systems has considerably increased due to the popularity of ontologies for the semantic web and specifically, because of the OWL language. The semantic web uses as standard the *OWL* language to structure knowledge contained in web sites, so, to describe ontologies. This language(OWL) has three variants, one of them is *OWL DL*, a language that uses the concepts of description logic we have explained in this section.

3.2 A DL Version of the Ontology

The ontology showed in Figures 2 and 3 defines a taxonomy of evaluative beliefs, that represents beliefs that have some social evaluations. We divided them into *SimpleBelief* and *MetaBelief*. This division is conceptually important when talking about cognitive agents. An agent holding a simple belief acknowledges the evaluation that the belief contains. This is not necessary in a Metabelief, since it is a belief about other agents beliefs, an interpretation of what other agents think. Therefore, an agent holding a Metabelief do not need to believe the nested evaluation. For instance, I can believe that my friend thinks that his car is nice, but I don't necessary agree with this opinion. Then, we consider an Image, Direct Experiences and a SharedVoice as simple beliefs, and Reputation and Shared Images as Metabeliefs (see [21] for the details of this decision). The meaning of these objects is the same with the ones we have described in Repage. A direct experience should be understood as an outcome predicate in Repage.

These concepts are located in the bottom part of the taxonomy. A system using this ontology will have instances of these concepts. All them have, at least, one attribute that is an object *Evaluation* containing information about the evaluation itself. Part of this information is the value of the evaluation, the representation of goodness and badness. In literature there are several possible representations, from simple boolean with bad/good, to probability distributions over some sorted set, like in the case of Repage. In [21] the authors describe four representation types, including transformation operations between them. For the sake of simplicity, in this first approach we will use a simple sorted labeled set, *VB, B, N, G, VG* meaning, *Very Bad*, *Bad*, *Neutral*, *Good* and *Very Good*.

At this point we have all the elements to understand a description of the ontology, that corresponds with the *TBox* of a DL system[2]:

[2] The semantics of $(\leq\ nR)$ and $(=\ nR)$ is defined as $(\leq\ nR)^I = \{a \in \triangle^I where |\{b|(a,b) \in R^I\}| \leq n\}$ and $(=nR)^I = \{a \in \triangle^I where |\{b|(a,b) \in R^I\}| = n\}$, where I is an interpretation, $\triangle^I$ the domain of the interpretation, and R^I the interpretation of the relation R.

$$
\begin{aligned}
Entity &\equiv SingleAgent \sqcup Group \sqcup Institution \\
Focus &\equiv Skill \sqcup Standard \sqcup Norm \\
Evaluation &\equiv \leq 1 hasSource.Entity \sqcap = 1 hasTarget.Entity \sqcap \\
&\equiv \sqcap = 1 hasContext.Focus \leq 1 hasStrength.\mathbb{R} \sqcap \\
&\equiv \sqcap \leq 1 hasValue.Value \\
Voice &\equiv \leq 1 hasGossiper.Entity \sqcap \leq 1 hasRecipient.Entity \sqcap \\
&\equiv = 1 hasEval.Evaluation \\
Image &\equiv SimpleBelief \sqcap = 1 hasEval.Evaluation \\
DExperience &\equiv SimpleBelief \sqcap = 1 hasEval.Evaluation \sqcap = 1 hasTrans.\mathbb{R} \\
ShVoice &\equiv SimpleBelief \sqcap = 1 hasVoice.Voice \sqcap \exists hasGossiper.Entity \\
ShImage &\equiv MetaBelief \sqcap = 1 hasEval.Evaluation \sqcap \exists hasSource.Entity \\
Reputation &\equiv MetaBelief \sqcap = 1 hasVoice.Voice
\end{aligned}
$$

In this case we consider as primitive concepts *SingleAgent*, *Group*, *Institution*, *Skill*, *Standard* and *Norm*. The concept *Value* is used to define the predicates *Value(VB)*, *Value(B)*, *Value(N)*, *Value(G)* and *Value(VG)*, as axioms of the system. All other concepts are defined using the $\mathcal{ALUN}$-language (see [22]).

4 The LRep: A Language for Reputation and Image Justification

In this section we define both the syntax and semantics of the LRep language. The objective of this language is to provide a mechanism to represent not only the evaluation of an image or reputation but also a justification of that value. This justification should increase the richness of the exchanged information about image and reputation and therefore, increase the effectiveness of spreading them. That justification can be sometimes even more relevant than the evaluation itself (see section 5).

First, we will define the syntax of the language giving an informal semantics. Finally, we will give a formal semantics of the language.

4.1 Defining the Basis of LRep

Let $A = \{a_1, \ldots, a_n\}$, $R = \{r_1, \ldots, r_m\}$ and $V = \{VB, B, N, G, VG\}$ be a set of agents, a set of roles, and a sorted set of evaluation labels respectively. We define the set *Eval* of all possible evaluations and evaluation as follows:

$$Eval = \{< a, r, v > | a \in A, r \in R, v \in V\} \tag{1}$$

We define a set of predicate letters P, and a set of quantifier letters N

$$P = \{I, R, ShI, ShV, DE, CI, CI_1, \ldots, CI_n, CR, CR_1, \ldots, CR_n\} \tag{2}$$

$$N = \{N_1, \ldots, N_n\} \tag{3}$$

Intuitively, the letters I, R, ShI and ShV refer to evaluations that are *Image*, *Reputation*, *Shared Image* and *Shared Voice*. The predicates CI, and CR refer to *Communicated Image* and *Communicated Reputation*. Concretely , CI_i and CR_i reefer to a *Communicated Image* and *Communicated Reputation* from an agent $a_i \in A$. DE refers to a *Direct Experience*. Notice that in the ontology, this predicate has an object evaluation and a real value. This second one refers to an identification number of the transaction that produced the direct experience.

4.2 Simple Predicate Formula (SPF) and Extended Predicate Formula (EPF)

Formulas in the LRep language are divided in SPF and EPF.

Let $e \in Eval$, $t \in \mathbb{R}$ and $1 \leq i \leq n$, then the following formulas are SPF:

- $I(e), R(e), ShI(e), ShV(e)$
- $DE(e,t), CI_i(e), CR_i(e)$

Let $1 \leq i \leq n$ and $e \in Eval$ then

- $\emptyset$ (empty formula) is an EPF
- If α is SPF then α is EPF
- If α is SPF then $N_i\alpha$ is EPF
- The formulas $N_iDE(e)$, $N_iCI(e)$ and $N_iCR(e)$ are EPF
- Inductively, if β and γ are EPF, then $\beta;\gamma$ is EPF

Intuitively, N_iX means that the agent has received at least i communicated images or communicated reputations, or that the agent has had at least i direct experiences[3]. The formal semantics of the quantifier is defined in Section 4.4. Also, we say that all formulas that are SPF as well as the formulas $N_iDE(e)$, $N_iCI(e)$ and $N_iCR(e)$ are atomic formulas.

4.3 Justification

We define a justification in terms of LRep language as follows. Let α be a SPF and γ be an EPF, then a LRep formula is defined as:

$$\{\alpha : \gamma\} \tag{4}$$

The idea is that in the expression $\{\alpha : \gamma\}$, the α predicate is the main element to communicate, and it is *justified* by the formula γ, that in fact it is a list of less generic predicates. For example we can have justifications like this:

$$\{I(< a_1, r_1, VB >) : N_5CI(< a_1, r_1, B >); N_3DE(< a_1, r_1, VB >); \\ CI_{a_3}(< a_1, r_1, VB >)\}$$

[3] We decide N_i to be a lower bound instead of an exact number because this second case is too restrictive and leads to only an honest-liars communication, forgetting the interesting option of telling a truth information but not exact. We have in mind to include in the future negative connective that will allow setting upper and lower bounds.

The above expression means that the Image of a_1 towards the role r_1 is very bad *because* we have received more than 5 communicated images saying that a_1 in r_1 is bad, we have experienced more that 3 times that the agent is very bad, and because a_3 communicated us that a_1 in the role r_1 is very bad. Of course, we are not talking about neither the truth of the explanation, nor the truth of the communication itself. Agents can lie, and of course can give partial information.

The syntax of LRep language can be defined using the following grammar.

$$
\begin{aligned}
LRep &::= \{SPF : EPF\} \\
SPF &::= I(E)|R(E)|ShI(E)|ShV(E)|DE(E, I\!N)|Comm \\
Comm &::= CI_{agent}(E)|CR_{agent}(E) \\
E &::= < Target, Context, Value > \\
EPF &::= \emptyset|N_{I\!N} CI(E)|N_{I\!N} CR(E)|N_{I\!N} DE(E)|SPF|EPF; EPF \\
Context &::= norm|standard|skill \\
Target &::= agent|group|institution \\
Value &::= VB|B|N|G|VG
\end{aligned}
$$

4.4 Semantics of LRep

To define the formal semantics of the language we have to introduce the concept of *correctness* within a LRep expression. Saying that I had more than 10 direct experiences with a seller when I really had 2 is not correct taking into account my state of the world (where I only had 2 direct experiences). So, the semantics of LRep will be determined for the correctness of the expression towards certain state of the world. Of course, this model of the world will be represented as an instance of a DL system with the *TBox* defined in Section 3.2.

So, let $F =< T, A >$ be a DL system describing the state of the world of an agent, where T is the TBox of terminological terms composed of the concepts defined in Section 3.2, and A the ABox with the assertions describing the state of the world at certain moment of time. We say that a justification $J = \{\alpha : \gamma\}$ is correct towards the system F, written as $F \supset J$ iff each of the components of J is correct towards F. More formally:

$$F \supset \{\alpha : \gamma\} \leftrightarrow F \supset \alpha \text{ and } F \supset \gamma$$

Then, the correctness of formulas SPF and EPF is defined in terms of the correctness of its atomic formulas. For instance, considering the atomic formula $DE(< y, r, v >, t)$ its correctness is defined as follows:

$$
\begin{aligned}
F \supset DE(< y, r, v >, t) \leftrightarrow \exists a, e \text{ such that} \\
F \models DExperience(a), hasEval(a, e), \\
hasTrans(a, t) \text{ and } evalFine(e, y, r, v, F)
\end{aligned}
$$

where we define the predicate $evalFine$ as follows:

$$
\begin{aligned}
evalFine(e, y, r, v, F) = True \leftrightarrow F \models hasTarget(e, y) \text{ and} \\
F \models hasContext(e, r) \text{ and} \\
F \models hasValue(e, v) \text{ and} \\
F \models Value(v), Focus(r), Entity(y)
\end{aligned}
$$

Following the same idea, the correctness of all atomic elements of LRep is defined in the next table:

$$F \supset I(< y, r, v >) \leftrightarrow \exists a, e \text{ such that } F \models Image(a), hasEval(a, e), \text{ and } evalFine(e, y, r, v, F)$$

$$F \supset CI_x(< y, r, v >) \leftrightarrow \exists a, e \text{ such that } F \models ShImage(a), hasSource(a, x), hasEval(a, e) \text{ and } evalFine(e, y, r, v, F)$$

$$F \supset CR_x(< y, r, v >) \leftrightarrow \exists a, v, e \text{ such that } F \models ShVoice(a), hasGossiper(a, x) \; hasVoice(a, v), hasEval(v, e) \text{ and } evalFine(e, y, r, v, F)$$

$$F \supset R(< y, r, v >) \leftrightarrow \exists a, v, e \text{ such that } F \models Reputation(a), hasVoice(a, v) \; hasEval(v, e) \text{ and } evalFine(e, y, r, v, F)$$

$$F \supset N_i DE(< y, r, v >) \leftrightarrow |A| \geq i \text{ where } A = \{DExperience(a) | \exists e, t \text{ such that } F \models hasEval(a, e), hasTrans(a, t) \text{ and } evalFine(e, y, r, v, F)\}$$

$$F \supset N_i CI(< y, r, v >) \leftrightarrow |A| \geq i \text{ where } A = \{Entity(x) | \exists a, e \text{ such that } F \models ShImage(a), hasSource(a, x) \; hasEval(a, e) \text{ and } evalFine(e, y, r, v, F)\}$$

$$F \supset N_i CR(< y, r, v >) \leftrightarrow |A| \geq i \text{ where } A = \{Entity(x) | \exists a, e \text{ such that } F \models ShVoice(a), hasGossiper(a, x) \; hasVoice(a, v), \; hasEval(v, e) \text{ and } evalFine(e, y, r, v, F)\}$$

Finally, let γ be an EPF formula, if $\gamma \equiv \emptyset$ then $F \supset \gamma$. If $\gamma \equiv \beta_1; \beta_2$ then, $F \supset \gamma \leftrightarrow F \supset \beta_1$ and $F \supset \beta_2$.

5 Using LRep

In this section we apply LRep in a concrete scenario. Let A be the set of agent names $A = \{John, Debra, Laura, \ldots\}$ and R a set of roles $R = \{seller, informant, buyer\}$. In this environment, everybody can play the three roles. In a typical transaction, an agent acting as a *buyer*, buys a specific product from another agent that acts as a *seller*. Also, there is the possibility to exchange information about other agents' performance, acting then as an *informant*. Agents are cognitive and use the Repage model to deal with social evaluations. In this case they evaluate agents as *sellers* (whether they sell the products with the

maximum quality, as they claim) and as *informants* (since they may not provide accurate information or even they may lie). Currently the exchange of social evaluations is done in terms of Image or Reputation. As shown in Section 2 there is an implicit commitment sending an Image (since it is the agent's own opinion) that does not exists when sending Reputation.

After introducing the scenario, we expose several cases where by using a justification, ambiguous situations become clearer and communications richer.

5.1 Case 1: Discrimination between Weak and Strong Predicates

One of the main issues when exchanging social evaluations is the inherent subjectivity that they are associated with. Check for instance, the following communications:

$$\begin{aligned}
&\text{C1:}\{I(< John, seller, VG >)\}\\
&\text{C2:}\{I(< John, seller, VG >) : N_2DE(< John, seller, VG >)\}\\
&\text{C3:}\{I(< John, seller, VG >) : N_{20}DE(< John, seller, VG >)\}
\end{aligned}$$

The first communication, $C1$, indicates that the image the informant has of *John* as a *seller* is VG (very good). However, it does not tell us anything about the strength of it. Communications $C2$ and $C3$ show us some more details. Assuming that agents send correct information towards its vision of the world (in the sense we define in Section 4.4), we should agree that the justification in $C3$ gives more reasons to belief the communicated image than $C2$. And in this sense, communicated image in $C3$ is *stronger* than the one in $C2$ and of course than the one in $C1$. In terms of reputation we can have similar situations.

$$\begin{aligned}
&\text{C1:}\{R(< John, seller, VG >)\}\\
&\text{C2:}\{R(< John, seller, VG >) : N_2CR(< John, seller, VG >)\}\\
&\text{C3:}\{R(< John, seller, VG >) : N_{20}CR(< John, seller, VG >)\}
\end{aligned}$$

5.2 Case 2: Avoiding Unreliable Information

Another case where the use of LRep helps in a better understanding of the messages, is in the detection of information that should not be taken into account because the justification contradicts the state of the world that the recipient has. For instance, check the following justification:

$$\begin{aligned}
\{I(< John, seller, B >) : &\ CI_{Laura}(< John, seller, B >);\\
&\ CI_{Debra}(< John, seller, VB >);\\
&\ I(< Laura, informant, VG >);\\
&\ I(< Debra, informant, VG >)\}
\end{aligned}$$

In this case, the informant justifies its image of *John* as a seller pointing out that he has received two communicated images, one from *Laura* and another from *Debra* (that are considered very good informants), saying that *John* is mostly bad. However, if the recipient of the message has an image of *Laura* and *Debra* as informants that is very bad the image of *John* cannot be considered, at least without further knowledge that could solve the contradiction.

5.3 Case 3: Control of Granularity

One interesting property that LRep has is the granularity of its predicates. In this sense, even in this first version it is already possible to give more and more detailed information to properly justify a communication. For instance, let's consider the following communication:

$$\{R(< Laura, seller, VG >) : ShV(< Laura, seller, G >); \\ ShV(< Laura, seller, VG >)\}$$

Here, this agent is justifying a reputation by means of two shared voices that at the same time are justified as follows:

$$\{ShV(< Laura, seller, G >) : N_1CR(< Laura, seller, G >)\} \\ \{ShV(< Laura, seller, VG >) : N_2CR(< Laura, seller, VG >)\}$$

Another possible and more detailed justification of the two shared voices could be:

$$\{ShV(< Laura, seller, G >) : \quad CR_{Debra}(< Laura, seller, G >)\} \\ \{ShV(< Laura, seller, VG >) : CR_{John}(< Laura, seller, VG >); \\ CR_{John}(< Jorge, seller, VG >)\}$$

Therefore, this justification could have included some information about the images of the informants, that supposedly are good. And these images, can be justified with the detail that the agent considers appropriate. The point of this discussion is to make the reader notice that using LRep, agents can reach the level of detail they want in the justifications.

5.4 Case 4: Putting Everything Together: Dialogs

Finally, extending LRep by allowing questions we can establish dialogs between two agents. In the following example we have agents A_1 and A_2 exchanging information. Initially, A_1 sends an image without any justification.

$$A_1 \rightarrow A_2 : \{I(< Laura, informant, VG >)\}$$

At this point, A_2 does not know A_1 very well, then it asks for more information:

$$A_2 \rightarrow A_1 : \{I(< Laura, informant, VG >)?\} \\ A_1 \rightarrow A_2 : \{I(< Laura, informant, VG >) : \\ CI_{Laura}(< Debra, seller, VB >); I(< Debra, seller, VB >) \\ CI_{Laura}(< John, seller, VG >); I(< John, seller, VG >)\}$$

Again A_2 is not satisfied. It wants to know how the images about *Debra* and *John* where formed, so, it asks for it:

$A_2 \rightarrow A_1 : \{I(< Debra, seller, VB >)?\}$
$A_2 \rightarrow A_1 : \{I(< John, seller, VG >)?\}$
$A_1 \rightarrow A_2 : \{I(< Debra, seller, VB >) : N_3DE(< Debra, seller, VB >)\}$
$A_1 \rightarrow A_2 : \{I(< John, seller, VG >) : N_2DE(< John, seller, VG >)\}$

Now, A_2 knows that the original information about *Laura* as *informant* is very good for A_1, because it is based on the following: *Laura* gave information about *Debra* and *John* as being very good and very bad sellers respectively. Furthermore, A_2 had an experience with both of them observing that they behaved in the same way that *Laura* said.

Knowing this, the conclusions that A_2 may get depend on its own state of the world, its beliefs:

- **Ignore the information:** It may have already some direct experiences with *Debra* and *John* and they behaved the opposite of what A_1 claims. In this case, the information that *Laura* as *informant* is very good is not reliable for A_2.
- **Take the information as reliable:** In this case, the evaluations of the direct experiences that A_2 and A_1 had with *Debra* and *John* may coincide, and then, A_2 may consider the original information reliable.
- **Need for more information:** Another case may come out when for instance, A_2 does not have any information about *John* or *Debra*. In this case, if for A_2 the original information is important enough and have the chance to do it, it may interact with both to acquire first hand experiences, or it may ask another agent (with good image as informant) to contrast the received information. The idea is that in justifications, every piece of information can be contrasted, either by direct experiences or by communications. So, in this example, the number of possible actions is quite high.

6 Conclusions and Future Work

As we stated from the beginning, we are dealing with cognitive agents. In our case it means agents that have beliefs, desires, intentions and goals to accomplish and that are able to reason about them. This is the context where a language like LRep has sense. By exchanging not only simple image/reputation values but justifications of these values, we are opening the possibility to reason about the process the informant followed to build those values and not only about the values themselves. Talking about image and reputation, and as we have shown with some examples in section 5, that extra information can be as useful as the value itself.

An important aspect of LRep is that the informant can decide how deep the justification has to be. Agents can choose from a wide range of possibilities when: From no justification at all to the exact details of the calculation. Furthermore, the fact of using a common ontology of reputation for the LRep semantics, allows to apply LRep in other reputation models.

Future experiments are planned to be done using Repage and LRep. In a scenario like the one described in [23], we have a set of buyers and sellers. Sellers sell items with certain quality (from a predefined minimum and maximum), and buyers want to buy always the maximum quality. Providing the agents with Repage system, in [23] several experiments were run to observe the performance of the buyers per turn, varying several parameters (like number of sellers or buyers) and dealing with cheaters. The incorporation of the LRep language in these simulations will require two parallel phases. On the one hand, design more sophisticated decision making processes to take advantage of this new functionality, and on the other hand, study the impact, the creation and motivation of sending false information in justifications.

Besides this, the LRep language is very simple, almost every atomic element in LRep coincides with an element of the ontology. Only the quantifiers define more sophisticate semantics. Extensions of LRep are expected, for instance, including universal or existential quantifiers. Also, more sophisticates protocols of communication should be taken into account.

Acknowledgments

This work was supported by the European Community under the FP6 programme (eRep project CIT5-028575 and OpenKnowledge project FP6-027253) and the project Autonomic Electronic Institutions (TIN2006- 15662-C02-01), and partially supported by the Generalitat de Catalunya under the grant 2005-SGR-00093. Jordi Sabater-Mir enjoys a Ramon y Cajal contract from the Spanish Government.

References

1. Bromley, D.B.: Reputation, Image and Impression Management. John Wiley, Chichester (1993)
2. Karlins, M., Abelson, H.I.: Persuasion, how opinion and attitudes are changed. Crosby Lockwood & Son (1970)
3. Buskens, V.: The social structure of trust. Social Networks 20, 265–298 (1998)
4. Plato: The Republic (370BC). Viking Press (1955)
5. Hume, D.: A Treatise of Human Nature (1737). Clarendon Press, Oxford (1975)
6. Marimon, R., Nicolini, J.P., Teles, P.: Competition and reputation. In: Proceedings of the World Conference Econometric Society, Seattle (2000)
7. Celentani, M., Fudenberg, D., Levine, D.K., Psendorfer, W.: Maintaining a reputation against a long-lived opponent. Econometrica 64(3), 691–704 (1966)
8. eBay: eBay (2002), http://www.eBay.com
9. Amazon: Amazon Auctions (2002), http://auctions.amazon.com
10. OnSale: OnSale (2002), http://www.onsale.com
11. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Proceedings of the Hawaii's International Conference on Systems Sciences, Maui, Hawaii (2000)

12. Esfandiari, B., Chandrasekharan, S.: On how agents make friends: Mechanisms for trust acquisition. In: Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada, pp. 27–34 (2001)
13. Schillo, M., Funk, P., Rovatsos, M.: Using trust for detecting deceitful agents in artificial societites. Applied Artificial Intelligence (Special Issue on Trust, Deception and Fraud in Agent Societies) (2000)
14. Yu, B., Singh, M.P.: Towards a probabilistic model of distributed reputation management. In: Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada, pp. 125–137 (2001)
15. Carbo, J., Molina, J., Davila, J.: Trust management through fuzzy reputation. Int. Journal in Cooperative Information Systems (2002) (in press)
16. Sen, S., Sajja, N.: Robustness of reputation-based trust: Boolean case. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS 2002), Bologna, Italy, pp. 288–293 (2002)
17. Sabater, J., Paolucci, M., Conte, R.: Repage: Reputation and image among limited autonomous partners. J. of Artificial Societies and Social Simulation 9(2) (2006)
18. Carter, J., Bitting, E., Ghorbani, A.: Reputation formalization for an information-sharing multi-agent sytem. Computational Intelligence 18(2), 515–534 (2002)
19. Sabater, J., Sierra, C.: Regret: A reputation model for gregarious societies. In: Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada, pp. 61–69 (2001)
20. Conte, R., Paolucci, M.: Reputation in artificial societies: Social beliefs for social order. Kluwer Academic Publishers, Dordrecht (2002)
21. Pinyol, I., Sabater-Mir, J., Cuni, G.: How to talk about reputation using a common ontology: From definition to implementation. In: Proceedings of the Ninth Workshop on Trust in Agent Societies, Hawaii, USA, pp. 90–101 (2007)
22. Baader, F., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The description logic handbook. Cambridge University Press, Cambridge (2003)
23. Pinyol, I., Paolucci, M., Sabater-Mir, J., Conte, R.: Beyond accuracy. reputation for partner selection with lies and retaliation. In: Proceedings of the Eighth International Workshop on Multi-Agent-Based Simulation, pp. 134–146 (2007)

From AO Methodologies to MAS Infrastructures: The SODA Case Study

Ambra Molesini[1,⋆], Enrico Denti[1], and Andrea Omicini[2]

[1] Alma Mater Studiorum—Università di Bologna
viale Risorgimento 2, 40136 Bologna, Italy
ambra.molesini@unibo.it, enrico.denti@unibo.it
[2] Alma Mater Studiorum—Università di Bologna a Cesena
via Venezia 52, 47023 Cesena, Italy
andrea.omicini@unibo.it

Abstract. In the last years, research on agent-oriented (AO) methodologies and multi-agent system (MAS) infrastructures has developed along two opposite paths: while AO methodologies have essentially undergone a top-down evolution pushed by contributions from heterogeneous fields like human sciences, MAS infrastructures have mostly followed a bottom-up path growing from existing and widespread (typically object-oriented) technologies. This dichotomy has produced a conceptual gap between the proposed AO methodologies and the agent infrastructures actually available, as well as a technical gap in the MAS engineering practice, where methodologies are often built *ad hoc* out of MAS infrastructures, languages and tools.

This paper proposes a new method for filling the gap between methodologies and infrastructures based on the definition and study of the meta-models of both AO methodologies and MAS infrastructures. By allowing structural representation of abstractions to be captured along with their mutual relations, meta-models make it possible to map design-time abstractions from AO methodologies upon run-time abstractions from MAS technologies, thus promoting a more coherent and effective practice in MAS engineering. In order to validate our method, we take an AO methodology – SODA – and show how it can be mapped upon three different MAS infrastructures using meta-models as mapping guidelines.

1 Introduction

Traditional software engineering approaches and metaphors fall short when applied to areas of growing relevance such as electronic commerce, enterprise resource planning, and mobile computing: such areas, in fact, generally call for open architectures that may evolve dynamically over time so as to accommodate new components and meet new requirements. This is probably one of the main reasons why the *agent* metaphor and the agent-oriented paradigm are gaining

⋆ Corresponding author.

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 300–317, 2008.

momentum in these areas. At the same time, such a rapid paradigm shift dropped technology behind: while in the past new abstractions used to come from programming languages, and were later included in software engineering practice, now it is often the case that technologies adopted for MAS development and deployment do not support the novel abstractions adopted in the agent-oriented software engineering (AOSE) analysis and design phases.

Such a gap mainly depends on AO methodologies and MAS infrastructures having evolved along two parallel, yet somehow inverse, paths: a top-down evolution for AO methodologies, a bottom-up path for multi-agent infrastructures. In fact, on the one side, abstractions and metaphors (models and structures) from human organisations have been used to analyse, model and design software systems, leading to methodologies like Gaia [1,2], Tropos [3,4], PASSI [5,6] and SODA [7,8]. There, modelling *agent societies* means to identify the global rules that should drive the expected MAS evolution, and the roles that agents should play. On the other side, MAS infrastructures have typically evolved out from existing (mainly, object-oriented) programming languages and development environments, "stretching" the agent paradigm on top of more traditional paradigms and technologies [9]. For instance, infrastructures such as TuCSoN [10,11], TOTA [12,13] and CArtAgO [14,15] introduce specific agent-oriented abstractions (tuple centres, co-fields, artifacts) to constructively constrain the design and final architecture of MAS: yet, the imprint of the object-oriented paradigm is still visible—for instance, in agents taking the form of Java threads. The above gap can lead to inconsistencies between the design and the actual implementation of a system, as the agent-based concepts and metaphors adopted in the analysis and design phases can not match the development tools used for system implementation and deployment, which are often in the stage of academic prototypes.

In this context, this paper is aimed at highlighting some guidelines for correctly mapping the abstractions adopted by an AO methodology onto the abstractions supported by MAS infrastructures: we assume SODA as a case study, and discuss how its design abstractions could be mapped onto three MAS infrastructures —TuCSoN, CArtAgO and TOTA. Accordingly, we first analyse and compare the meta-models of the SODA methodology and of the chosen infrastructures, then exploit them to express both the structural representation of the elements constituting the actual system, and their relationships [16].

Accordingly, the paper is structured as follows. Section 2 sketches a possible classification of AO methodologies based on their relations with MAS technologies, and highlights the main advantages of their meta-model representation. Then, Section 3 discusses the meta-models of the SODA methodology and of the selected infrastructures, whereas Section 4 presents the mapping of SODA abstractions onto such infrastructures and discusses the key guidelines. Section 5 reports an example of our approach mapping the SODA's design onto a TuCSoN-based implementation. Related work, conclusions and future works are reported in Section 6 and Section 7.

2 AOSE Methodologies: Technologies and Meta-models

MAS are a powerful paradigm for the implementation of complex computational systems: the aim of AOSE is to effectively support the path from (an agent-oriented) design to (an agent-based) deployment of the system. This is why methodologies (and respective notations) have become central in AOSE research as a key tool in the MAS analysis, design and development process.

Among the current methodologies, some are rooted in artificial intelligence (AI), others emerge as an extension of object-oriented (OO) methodologies, further try to merge the two approaches in some original way; finally, others are not directly derived by previous approaches. So, an important classification criterion is to distinguish methodologies that are neutral with respect to the implementation technologies (*technology-neutral methodologies*, or simply *neutral* ones in the following), from those that are bound to specific infrastructures (*technology-biased methodologies*, or simply *biased* ones in the following)—usually, due to the choice of developing a CASE tool for supporting rapid prototyping and code generation [17,18]. The first category includes methodologies like Gaia [1,2], MESSAGE [19,20], INGENIAS [21], and SODA [7,8]: they all aim at guiding the designer from the requirement analysis phase down to the design phase, yet with no assumptions on the implementation and deployment phases—probably, because of the lack of a recognised standard language and infrastructure that could natively support agent-oriented concepts. Among biased methodologies, Tropos [3,4] and PASSI [5,6] are both tied to the JADE [22,23] infrastructure, and come with a set of development support tools.

Since there is currently no widely-acknowledged standard infrastructure for MAS implementation, it is unclear whether committing to an infrastructure at the methodological level may be better or worse than opting for technological neutrality. In fact, even though neutral methodologies suffer from a deeper gap with respect to the underlying technology, their models are general enough to be potentially implemented over any infrastructure by just providing suitable guidelines for mapping methodology abstractions onto infrastructure ones.

Apart from the technology neutrality matter, however, all AO methodologies introduce some basic abstractions (agents, roles, behaviour, ontology, ...) organised in a set of independent – but strongly correlated – models and phases. The relationships between such entities and the models can then be expressed by means of a *meta-model* [24,25], which becomes the key tool to compare methodologies with each other, identify families of (related) methodologies, and check the consistency of a methodology when planning extensions or modifications. So, a well-defined meta-model should address several different methodological aspects—for instance, the process to be followed, the work products to be generated, who is responsible for each process phase (analysts, designers, ...), etc.

Meta-models are also an important guide for integrating different methodologies avoiding several errors [24], such as assuming that differences of concern exist when none exists, or assuming similarity of concern because of a common use of terms—despite a different semantics. In the same way, infrastructure meta-models associate each methodology concept to some suitable infrastructural

abstraction. So, we expect that studying and comparing methodologies with infrastructures in terms of meta-models makes it possible to provide guidelines for mapping the design model of a methodology onto its implementation.

3 Meta-models for the Case Study

In the following we first introduce and analyse the meta-models of of the SODA methodology (Subsection 3.1) and of the three selected infrastructures—TuCSoN (Subsection 3.2), CArtAgO (Subsection 3.3), and TOTA (Subsection 3.4).

Then (Section 4) we discuss the guidelines for mapping the SODA concepts and metaphors onto such infrastructures, based on their meta-models.

3.1 SODA

SODA (Societies in Open and Distributed Agent spaces) [7,8,26,27] is an agent-oriented methodology for the analysis and design of agent-based systems. Since the original version [7], SODA has always focused on inter-agent issues, like the engineering of agent societies and the environment for MAS: in this perspective, it has recently been reformulated according to the A&A meta-model [28,29,30], where artifacts take the form of computational devices that populate the agent environment, and provide some kind of function or service used by agents [28]. Agents are used to model individual activities, while artifacts shape the MAS environment [29]. More generally, artifacts make it easier to enrich the MAS design with social and organisational structures, as well as with complex security models: roles, permissions, policies, commitments, and the like can be represented explicitly as first-class entities, and encapsulated within suitable artifacts. Many sorts of artifacts are supported by SODA, even if in the meta-model we refer to them simply as "artifact" without specify their typology. In particular, artifacts used to mediate between individual agents and the MAS are called *individual artifacts*, whereas *social artifacts* build up agent societies, and *environmental artifacts* mediate between the MAS and an external resource [29].

SODA is organised in two phases, each structured in two sub-phases: the *Analysis phase*, which is composed of the Requirements Analysis and the Analysis steps, and the *Design phase*, which is composed of the Architectural Design and the Detailed Design steps. The meta-model that represents the abstract entities adopted by SODA is depicted in Figure 1.

Requirement Analysis. Several abstract entities are introduced for requirement modelling (see Figure 1 "requirement analysis" part): in particular, *requirement* and *actor* are used for modelling the customers' requirements and the requirement sources, respectively, while the *external-environment* notion is used as a container of the *legacy-systems* that represent the legacy resources of the environment. The relationships between requirements and legacy systems are then modelled in terms of suitable *relation* entities.

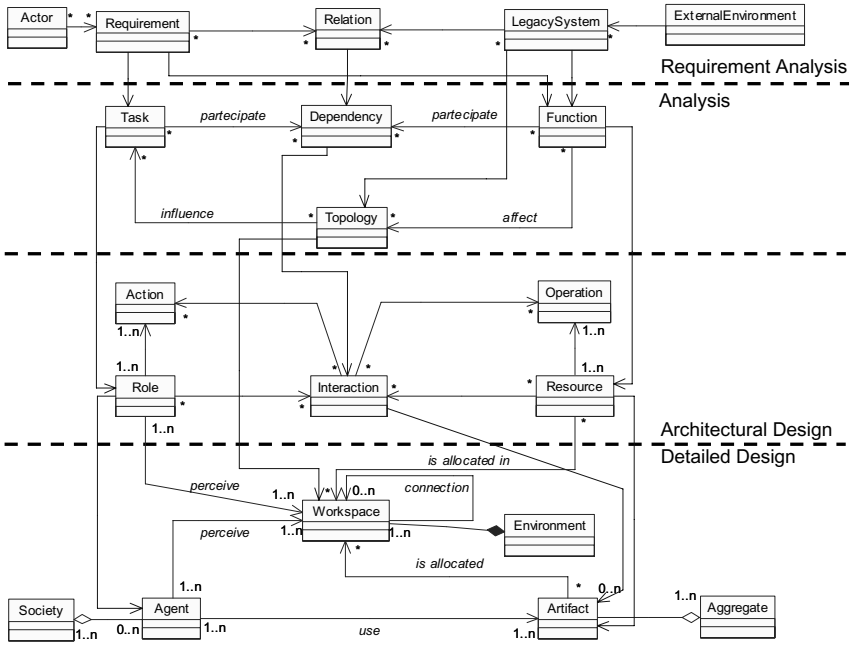


Fig. 1. SODA Meta-model

Analysis. The Analysis step expresses the abstract requirement representation in terms of more concrete entities such as *tasks* and *functions* (see Figure 1, "analysis" part). Tasks are activities requiring one or more competences, while functions are reactive activities aimed at supporting tasks. The relations highlighted in the previous step are now the starting point for the definition of *dependencies* (interactions, constraints, etc.) among the abstract entities. The structure of the environment is also modelled in terms of *topologies*, i.e. topological constraints over the environment. Topologies are often derived from functions, but can also constrain / affect task achievement.

Architectural Design. The main goal of this stage is to assign responsibilities of achieving tasks to *roles*, and responsibilities of providing functions to *resources* (see Figure 1, "architectural design" part). To this end, roles should be able to perform *actions*, and resources should be able to execute *operations* providing one or more functions. The dependencies identified in the previous phase become here *interactions*, i.e. "rules" enabling and bounding the entities' behaviour. Finally, the topology constraints lead to the definition of *workspaces*, i.e. conceptual places structuring the environment.

Detailed Design. Detailed Design is expressed in terms of *agents*, agent *societies*, *artifacts* and *artifact aggregates* (see Figure 1 "detailed design" part). Agents are intended here as autonomous entities able to play several roles, while societies are defined as the abstractions responsible for a collection of agents. The

resources identified in the previous step are now mapped onto suitable artifacts (intended as entities providing some services), while aggregates are defined as the abstractions responsible for a collection of artifacts. The *workspaces* defined in the Architectural Design step take now the form of an open set of artifacts and agents – that is, artifacts can be dynamically added to or removed from workspaces, as well as agents can dynamically enter (join) or exit workspaces.

3.2 TuCSoN

TuCSoN (Tuple Centres Spread Over Networks) [10,11] is an infrastructure providing services for the communication and coordination of distributed / concurrent independent agents: its meta-model is depicted in Figure 2.

In detail, TuCSoN supports agent communication and coordination via *tuple centres* coordination media [31]: these are shared & reactive information spaces, distributed over the infrastructure *nodes*. In turn, this inducts a *topology* over the *network*. Agents access tuple centres associatively, by writing (`out`), reading (`rd`, `rdp`), and consuming (`in`, `inp`) *tuples* — i.e., ordered collections of heterogeneous information chunks — via the above coordination primitives.

A tuple centre is a tuple space enhanced with the notion of behaviour specification. More precisely, a tuple centre is a coordination abstraction perceived by the interacting entities as a standard tuple space [32], but whose behaviour in response to events can be defined so as to embed the coordination laws. So, defining a new behaviour for a tuple centre basically amounts at specifying state transitions in terms of *reactions* to *events* [10]. In particular, reactions are specified in TuCSoN via the ReSpecT (Reaction Specification Language) language [30]: a reaction is defined as a set of non-blocking operations [10], and has a success/failure transactional semantics: a successful reaction may atomically produce effects on the tuple centre state, a failed reaction yields no result at all. Typically, a tuple centre contains a set of reactions (*reaction spec* in Figure 2),

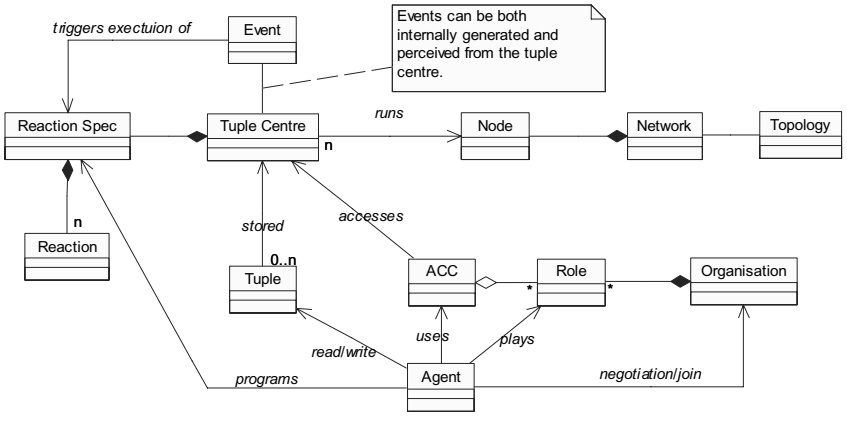


Fig. 2. TuCSoN Meta-model

each tied to a specific event: the same event could trigger multiple, different reactions. Tuple centres are connected each other through *link* operations, having the same form and a similar semantics as TuCSoN coordination primitives, but invoked by successful reactions rather than by agents [30].

The Agent Coordination Context (*ACC*), introduced in [33] as the conceptual place where to set the boundary between the agent and the environment, encapsulates the interface enabling agent actions and perceptions inside the environment. More precisely, an ACC (*i*) works as a model for the agent environment, by describing the environment where an agent can interact, and (*ii*) enables and rules the interactions between the agent and the environment, by defining the space of the admissible agent interactions. The ACC dynamics is characterised by two basic steps: *negotiation* and *use*. In fact, an ACC is meant to be first negotiated by the agents with the MAS infrastructure, in order to start a working session inside an *organisation*. To this end, the agent specifies which *roles* to activate: if the agent request is compatible with the (current) organisation rules, a new ACC is created, configured according to the characteristics of the specified roles, and is released to the agent for active playing inside the organisation. The agent can then use the ACC to interact with other agents in the organisation, and with the organisation environment, by performing the actions and activating the perceptions made possible by the ACC.

3.3 CArtAgO

The abstract architecture of CArtAgO (Common Artifact for Agents Open environment) [14,15] is composed of three main elements (see Figure 3): (*i*) *agent bodies* – as the entities that make it possible to situate agents inside the working

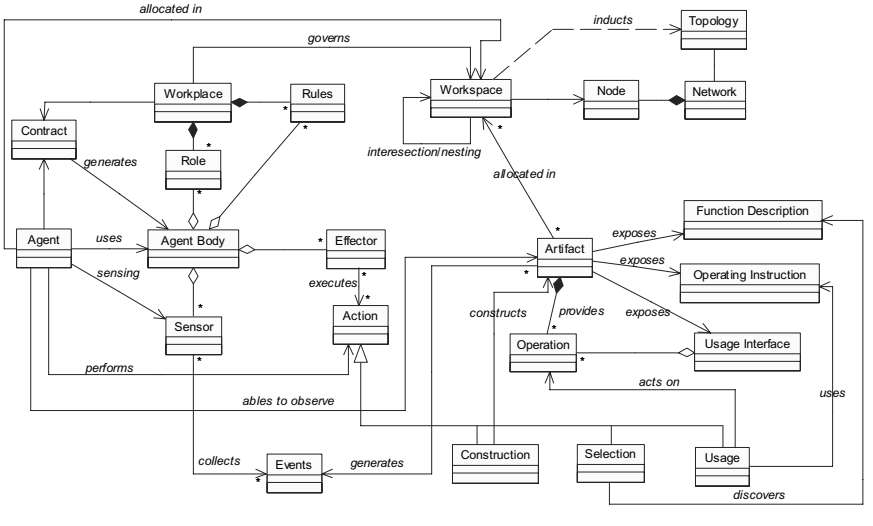


Fig. 3. CArtAgO Meta-model

environment; (*ii*) *artifacts* – as the basic building blocks to structure the working environment; and (*iii*) *workspaces* – as the logical containers of artifacts, aimed at defining the topology of the working environment.

Agent bodies. The agent body contains *effectors* to perform *actions* upon the working environment, and a dynamic set of *sensors* to collect *events* from the working environment. Agents interact with their working environment by "piloting" their bodies: they execute actions to *construct*, *select* and *use* artifacts, and perceive the *observable events* generated by artifacts.

Artifacts. Artifacts are the basic bricks managed by CArtAgO: agents use artifacts by triggering the execution of *operations* listed in the artifact *usage interface*. The execution of an operation typically causes the update of the internal state of an artifact, and the generation of one or more observable events: these are then collected by the agent sensors and perceived by means of explicit sensing actions. In order to support a rational exploitation of artifacts by intelligent agents, each artifact is equipped with a *function description*, i.e. an explicit description of the functionalities it provides, and *operating instructions*, i.e. an explicit description of how to use the artifact to get its function.

Workspaces. Artifacts are logically located in workspaces, which define the *topology* of the working environment. A workspace is an open set of artifacts and agents: artifacts can be dynamically added to or removed from workspaces by agents, agents can dynamically enter (join) or exit workspaces. Workspaces make it possible to structure agents and artifacts organisation & interaction: in particular, workspaces can function as scopes for event generation and perception, as well as for artifact access and use. Articulated topologies can be created via workspace intersection and nesting: in particular, intersection is supported by allowing the same artifacts and agents to belong to different workspaces.

Workplaces, roles & contracts. In addition, CArtAgO also introduces the concept of *workplace* as an organisational layer on top of workspaces. More precisely, a workplace is the set of *roles* and *organisational rules* being in force in a workspace: there, *contracts* define the norms and policies that rule agent access to artifacts and allow the generation of agent bodies. So, for instance, an agent may or may not be granted permission to use some artifacts or to execute some specific operations on selected artifacts depending on the role(s) that the agent is playing inside the workplace [34].

3.4 TOTA

TOTA (Tuples On The Air)[12,13] is a middleware for multi-agent coordination, in distributed computing scenarios. A meta-model of the infrastructure is presented in Figure 4. TOTA assumes the presence of a network of possibly mobile *nodes*, each running a *tuple space* [31]: each agent is supported by a local middleware and has only a local (one-hop) perception of its environment. Nodes are connected only by short-range *network* links, each holding references to a

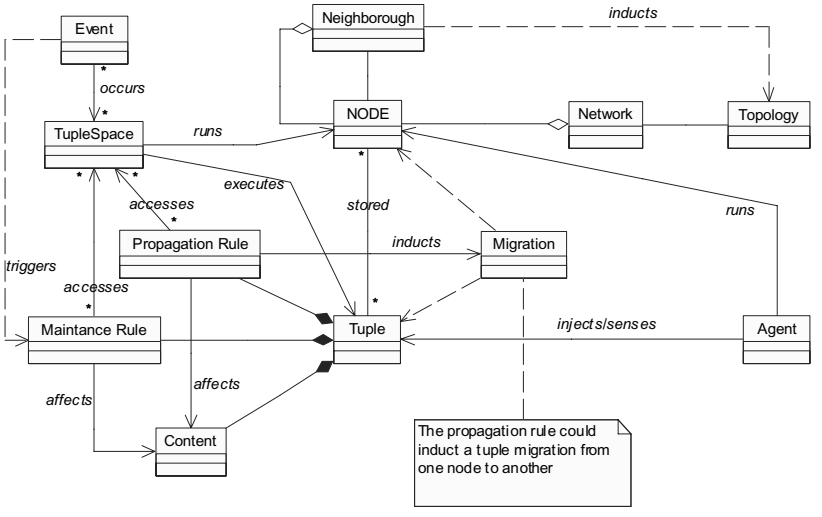


Fig. 4. TOTA Meta-model

(limited) set of *neighbour* nodes: so, the *topology* of the network, as determined by the neighbourhood relations, may be highly dynamic.

In TOTA, tuples are not associated to a specific node (or to a specific data space) of the network: rather, they are "injected" in the network by an agent from some node, then autonomously propagate hop-by-hop, diffuse, and evolve according to specified propagation patterns. Thus, TOTA tuples form a sort of spatially-distributed data structure, that can be used to acquire contextual information about the environment and to support the mechanisms required for stigmergic interaction [35]. More precisely, TOTA distributed tuples T=(C,P,M) are characterised by a *content* C, a *propagation rule* P, and a *maintenance rule* M: the content C is an ordered set of typed fields representing the information carried by the tuple, the propagation rule P determines how the tuple propagates across the network (called "migration" in the Figure 4) and how the tuple content should change while the tuple is propagated; finally, the maintenance rule M determines how a tuple distributed structure should react to *events* occurring in the environment. Specifying the tuple propagation rule includes determining the "scope" of the tuple and how such propagation is affected by the presence or absence of other tuples in the system. In turn, events handled by the maintenance rule can range from simple time alarms, to changes in the network structure: the latter kind of events is of fundamental importance to preserve a coherent structure of the environment properties represented by tuple fields.

4 From SODA to Infrastructures

This section presents some guidelines for mapping SODA design-level abstractions onto the infrastructural abstractions of TuCSoN (Subsection 4.1), CArtAgO

(Subsection 4.2) and TOTA (Subsection 4.3): the abstractions used in SODA analysis phase are left aside, as they would be too high-level with respect to infrastructures. Among the many MAS infrastructures available in literature, we choose TuCSoN and TOTA because interaction – and coordination, in particular – is at the core of both infrastructures, in the same way as in SODA. In addition, both infrastructures are not FIPA-compliant, and here we meant to explore such a sort of infrastructures. Finally we choose CArtAgO because it is the only infrastructure that natively supports the concepts of artifacts.

Such infrastructures are then compared so as to evaluate their support to SODA abstractions (Subsection 4.4).

4.1 SODA and TuCSoN

Since SODA is defined on top of the A&A meta-model, the first step is to define how agents and artifacts can be represented as TuCSoN abstractions.

Mapping the agent notion is straightforward, given TuCSoN native support for this concept: so there is a one-to-one mapping between SODA agent abstraction and TuCSoN one. However, the concept of agent action, explicitly considered in the SODA meta-model, is more or less reduced to the notion of coordination primitives, as performed by agents.

Mapping the artifact notion, instead, is less obvious, as SODA defines three different artifact types – social, individual and environmental artifacts – each requiring its own mapping. With respect to this issue, TuCSoN tuple centres can be seen as a special case of social artifacts: they mediate and govern agent interaction by encapsulating the laws of agent coordination. Such coordination laws, expressed in terms of reactions to interaction events, are well suited to map SODA interactions—i.e., the rules that enable and bound the entities' behaviour. In turn, the notion of individual artifact can be mapped onto the TuCSoN ACC concept, since its purpose is precisely to represent the interface of an agent towards the environment [36]. In fact, agents ask for an ACC specifying the roles to be activated: the ACC is then negotiated with the infrastructure as the agent joins the MAS organisation. If the negotiation is successful, the ACC is created and released to the agent, which, henceforth, exploits it to access the MAS services: the ACC redirects the agent invocations to the other artifacts in the environment. Finally, the environmental artifact is not natively supported by TuCSoN, so it must be developed if/when needed. Also, the notion of artifact operation is reduced here to the notion of tuple centre operation, and has not the generality required. Given that, link operations through tuple centres are the way in which TuCSoN artifacts are somehow composed.

Widening the view, the organisation concept provided by TuCSoN is well suited to represent SODA societies, in the same way as the TuCSoN role concept can well represent the SODA role notion. From the topological viewpoint, the SODA notion of workspace may be mapped onto the TuCSoN node concept, which, indeed, represents an open set of agents, tuple centres and ACCs; as a consequent step, TuCSoN network can be used to map SODA environments. On

the other hand, workspace connection, as introduced by SODA, has no mapping in TuCSoN, so it should be developed ad hoc when needed.

4.2 SODA and CArtAgO

CArtAgO and SODA share the same root in the A&A meta-model: so, quite expectedly, CArtAgO abstractions can easily support all SODA concepts. In particular, CArtAgO provides the artifact notion as a first-class abstraction, which can be used and easily specialised in social and environmental artifacts according to the developer's needs. Therefore, unlike TuCSoN, the SODA notion of artifact operation is directly mapped onto the operation abstractions supported by CArtAgO; the same holds for SODA agent action. Moreover, individual artifacts can be more specifically mapped on CArtAgO agent body abstraction, instead of using the generic artifact notion.

Composition of artifacts can also be easily realised, thanks to the *linkability* property [29] natively supported by CArtAgO artifacts to scale up with environment complexity. So, an artifact can be conceived and implemented as a composition of linked, possibly non-distributed, artifacts – or, conversely, a set of linked artifacts, scattered through a number of different physical locations, can be seen altogether as a single distributed artifact.

In addition, SODA organisational structure, which is defined in terms of roles and societies, can be easily translated on CArtAgO roles and workplaces. This makes it possible to capture SODA interaction concept in a straightforward way: in fact, interactions in SODA are aimed at enabling and constraining agent behaviour, which is precisely what the workplace rules and contract do – a CArtAgO agent may or may not have the permission to use some artifacts or to execute some specific operations on some specific artifacts depending on the role(s) that the agent itself is playing inside the workplace.

Finally, CArtAgO workspace concept can be directly used to map the SODA workspace concept, in the same way as CArtAgO workspace nesting supports SODA workspace connection. The CArtAgO abstractions of node, network and topology can be used to represent the SODA environment, too.

4.3 SODA and TOTA

TOTA provides a native support to the agent concept, while the artifact concept is supported only in the case of social artifacts. So, SODA agents can be directly mapped onto TOTA agents, while social artifacts are mapped onto TOTA tuple spaces. Unlike tuple centres, tuple spaces provide only a fixed coordination service: so, they are unable to support the SODA interaction concept. However, SODA social rules can be mapped onto the maintenance rule and the propagation rule associated to TOTA distributed tuples, exploiting the fact that propagation rules determine how tuples propagate through the network, and maintenance rules determine how the tuple distributed structure reacts to environment events. Of course, this mapping is less straightforward than in TuCSoN (whose reactions map SODA interaction concept directly): indeed, a set of many tuples

must be used to describe a single SODA interaction – each tuple representing one propagation and one maintenance rule. As a side effect of this one-to-many mapping, maintaining coherency is quite a hard task, and the rules/interaction mapping can often be very dispersive.

From the topology viewpoint, TOTA node concept maps SODA workspace concept: each node holds references to a limited set of neighbour nodes, and neighbourhood relations express the network topology. Such inter-node relations can be exploited also to provide an abstraction for mapping the SODA workspace connection concept. Finally, the TOTA network concept maps the SODA environment, too. All the others SODA concept are not natively supported by TOTA, and should therefore be developed in an *ad hoc* way when needed.

4.4 Discussion

Figure 5 highlights the SODA abstractions that are supported natively from each of the three infrastructures. The agent and resource abstractions are both omitted – the first because it is explicitly supported by each infrastructure, the latter for the opposite reason.

Quite expectedly, CArtAgO provides the best support for SODA design abstractions, as they are both rooted in the A&A meta-model: in particular, both consider the environment as the key element, adopt artifacts as their basic building blocks for modelling the environment resources, and workspaces for structuring the environment. Moreover, both SODA and CArtAgO support the MAS organisational structure by explicitly enabling the specification of social rules.

TuCSoN and TOTA, instead, provide support for fewer SODA abstractions: so, the developer needs to implement by himself the abstractions which are not supported by the infrastructure natively. In particular, none of the two infrastructures supports environmental artifacts, while both support social artifacts: this is not surprising, since they take both inspiration from coordination models, where interaction is typically mediated by some coordination media [31] (like

SODA	TuCSoN	CArtAgO	TOTA
Role	*Role*	*Role*	-
Action	*Coordination Primitive*	*Action*	-
Interaction	*Reaction Specification Reaction*	*Rules Contract*	*Maintenance Rule Propagation Rule*
Operation	*Tuple Centre Operation*	*Operation*	-
(Social) Artifact	*Tuple Centre*	*Artifact*	*Tuple Space, Tuples*
(Individual) Artifact	*ACC*	*Agent Body*	-
(Environmental) Artifact	-	*Artifact*	-
Aggregate	*Linked Tuple Centres*	*Artifact*	-
Society	*Organisation*	*Workplace*	-
Workspace	*Node*	*Workspace*	*Node*
Workspace Connection	-	*Workspace Nesting*	*Neighbourhood*

Fig. 5. Abstractions Mapping

a tuple space or a tuple centre) that could be easily seen as a special case of social artifact. Individual artifacts, in their turn, find their counterpart only in TuCSoN — namely, in the ACC abstraction. Moreover, SODA interaction abstraction, which represents the rules that enable and shape the agent behaviour, can be expressed directly by TuCSoN reactions, and indirectly via TOTA maintenance and propagation rules. Finally, as far as the organisational structure of the MAS is concerned, TuCSoN provides explicit abstractions such as organisation, role and ACC; on the other hand, TOTA does not provide any support for this issue yet, so the developer must provide for managing the MAS organisations on his/her own.

5 SODA and TuCSoN: An Example

In order to provide a concrete example of the effectiveness of our approach, we present a sketch of the mapping from SODA to the TuCSoN infrastructure: the complete case study can be found in [8]. Let us consider a Conference management system [2]. In the Architectural Design step, three main roles are identified (PC-chair, Author and PC-member): these can be mapped onto three corresponding TuCSoN roles played by three different TuCSoN agents. Moreover, in the Detailed Design step, each SODA agent is associated to an individual artifact (PC-Chair Artifact, Author Artifact and PC-Member Artifact): so, three TuCSoN ACCs have to be considered — one for each TuCSoN agent. Correspondingly, the actions associated to roles (executed by agents) have to be mapped onto suitable coordination primitives: for instance, the "write review" action, that allows PC-member to write the reviews, can be mapped onto the "out" coordination primitive [37].

On the other hand, since SODA's environmental artifacts are not supported by TuCSoN, these abstractions must be created *ad hoc* by developers — for instance, exploiting TuCSoN tuple centres as interfaces for standardising the access protocols to all the system resources. Then, as above, the operations associated to the resources (and performed by artifacts) can be mapped onto suitable tuple centre operations: for instance, the *store review* operation, that supports the *write review* action, can be mapped as:

```
paperArt?out(review(paper_id, rev))
```

where `paperArt` is the name of the tuple centre that stores the information about papers, and `review` is the name of the tuple posted in the tuple centre. The tuple arguments are the paper id (`paper_id`) and the review value (`rev`) decided by the PC-member [30].

The interactions individuated in the Architectural Design step, and associated to the social artifacts in the Detailed Design step, are consequently mapped onto suitable reactions. For example, the PC-member's action of downloading a paper for review is potentially critical from the fairness viewpoint, since he/ she could be one of the paper's authors: to face this issue, the action is subject to the *Review-Rule* managed by the `paperArt` social artifact. So, the PC-member just invokes the operation:

```
paperArt?in(download(paper_id, Paper_link))
```

on the paperArt tuple centre, which, in turn, triggers the reactions:

```
reaction( in(download(Paper_id, Paper_link)),
  (request, from_agent), (
    current_source(Agent),
    rd(association(Agent, PC_member)),
    rd(authorised(PC_member, Paper_id)),
    rd(link(Paper_id, Link)),
    out(download(Paper_id, Link))
)).

reaction( in(download(Paper_id, Paper_link)),
  (request, from_agent), (
    current_source(Agent),
    rd(association(Agent, PC_member)),
    no(authorised(PC_member, Paper_id)),
    out(download(Paper_id, nil))
)).
```

These perform the proper checks and either provide the PC-member access to the paper – by emitting the download(*paper_id*, *link*) tuple – or negate it—by emitting the download(*paper_id*,nil) tuple.

From the organisational viewpoint, the conference management system can be seen in SODA as an agent society, which is naturally mapped onto a TuCSoN organisation responsible for the creation and management of the ACCs representing the individual artifacts. Finally, since the SODA design specifies only one workspace, a single TuCSoN node seems enough for the purpose.

6 Related Work

Model-Driven Architecture [38] (MDA) is another approach for filling the gap among methodologies and infrastructures: its basic idea is to define first a Platform Independent Model (PIM) and then iteratively make it more and more platform-specific by a series of transformations, whose endpoint is the Platform Specific Model (PSM). Current technologies, however, may not fully support MDA complex transformation rules: for instance, UML, which is one of MDA foundations, lacks the required precision and formalisation [39].

Further research efforts are being devoted to integrating MDA and AOSE [39,40]. In [39], for instance, an agent architecture based on the human cognitive model of planning, the Cognitive Agent Architecture (Cougaar), is integrated with MDA. The resulting Cougaar MDA defines the models to be used, how they should be prepared, and the relationships among them. The level of application composition is thus elevated from individual components to domain-level model specifications in order to generate software artifacts. The software artifacts generation is based on a meta-model: each component is mapped onto

a UML structured component which is then converted into multiple artifacts—Cougaar/Java code, documentation, and test cases. In [40], Amor et al. show how the Model Driven Architecture (MDA) can be used to derive agent implementations from agent-oriented designs, independently from both the methodology and the concrete agent platform. Their goal is to study how to bridge the gap between methodologies and infrastructures, so as to cover the whole MAS lifecycle. Authors show how this problem can be naturally expressed in terms of MDA, and how MDA mechanisms can be used for defining the mappings. By applying the MDA ideas, the design model obtained from an agent-oriented methodology can be considered as a PIM, the target MAS agent platform as the PSM, and the mappings between the two can be given by the transformations defined for the selected agent platform. The target models need to be expressed in terms of their corresponding UML profiles, as indicated by MDA.

Since both methods imply the use of UML, its practical application requires that the selected AOSE methodology adopts UML or AUML as its notation: if this is not the case, like for many AOSE methodologies, an additional transformation from the methodology own notation to UML is necessary. As a result, the overall application of this approach involves many transformations for each mapping, and requires a PSM to be defined for each infrastructure.

7 Conclusions and Future Work

In this paper we adopted the SODA methodology as a running example for mapping the methodological concepts onto infrastructural abstractions in the case of three main agent infrastructures—TuCSoN, CArtAgO, and TOTA. To this end, we first studied the agent-oriented methodologies from the point of view of the connection with the implementation technologies, and classified them into *technology-neutral* and *technology-biased* methodologies.

Starting from neutral methodologies, that currently seem more appealing because of their independence from the underlying non-standard technologies, we then exploited meta-modelling as a tool to formalise the inner structure and the composing relationships both for the methodology and the selected infrastructures. Accordingly, we developed and comparatively analysed the meta-models of the SODA methodology and of CArtAgO, TuCSoN, and TOTA infrastructures, with double purpose of *(a)* providing guidelines for bridging the design and implementation phases, and *(b)* evaluating the quality of the mapping of SODA concepts onto infrastructural abstractions in terms of naturalness, clearness, and directness of the mapping. Finally we presented an example of our approach mapping the SODA's design of a case study onto a TuCSoN-based implementation.

Of course, this research is still in its early stage, so a lot of work remains to do: the next steps will be devoted to develop meta-models for other MAS infrastructures such as MARS [41], RETSINA [42] and JADE [22,23], and to

study how to map SODA concepts onto these infrastructures. In order to test our method, we also plan to make the same experiments taking as a base another neutral methodology, among Gaia [2], MESSAGE [19] or INGENIAS [21].

Acknowledgements

This work has been supported by the *MEnSA* project (*Methodologies for the Engineering of complex software Systems: Agent-based approach*) funded by the Italian Ministry of University and Research (MUR) in the context of the National Research 'PRIN 2006' call.

References

1. Zambonelli, F., Jennings, N., Wooldridge, M.: Multiagent systems as computational organizations: the Gaia methodology. In: [43], ch. VI, pp. 136–171
2. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) 12, 317–370 (2003)
3. Giorgini, P., Kolp, M., Mylopoulos, J., Castro, J.: Tropos: A requirements-driven methodology for agent-oriented software. In: [43], ch. II, pp. 20–45
4. Tropos: Home page, http://www.troposproject.org/
5. Cossentino, M.: From requirements to code with the PASSI methodology. In: [43], pp. 79–106
6. Cossentino, M., Sabatucci, L., Chella, A.: Patterns reuse in the PASSI methodology. In: Omicini, A., Petta, P., Pitt, J. (eds.) ESAW 2003. LNCS (LNAI), vol. 3071, pp. 294–310. Springer, Heidelberg (2004)
7. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 185–193. Springer, Heidelberg (2001)
8. SODA: Home page, http://soda.alice.unibo.it
9. Omicini, A., Rimassa, G.: Towards seamless agent middleware. In: IEEE 13th Inter. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2004). 2nd Inter. Workshop "Theory and Practice of Open Computational Systems" (TAPOCS 2004), pp. 417–422. IEEE Computer Society Press, Los Alamitos (2004)
10. Omicini, A., Zambonelli, F.: Coordination for Internet application development. Autonomous Agents and Multi-Agent Systems 2, 251–269 (1999)
11. TUCSON: Home page at SourceForge, http://tucson.sourceforge.net
12. Mamei, M., Zambonelli, F.: Programming stigmergic coordination with the TOTA middleware. In: Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M.P., Wooldridge, M. (eds.) Proceedings of AAMAS 2005, pp. 415–422. ACM Press, New York (2005)
13. Mamei, M., Zambonelli, F.: Programming modular robots with the tota middleware. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) 5th Inter. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 485–487. ACM Press, New York (2006)

14. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: A framework for prototyping artifact-based environments in MAS. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 67–86. Springer, Heidelberg (2007)
15. CARTAGO: Home page, http://cartago.alice.unibo.it
16. Molesini, A., Denti, E., Omicini, A.: MAS meta-models on test: UML vs. OPM in the SODA case study. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 163–172. Springer, Heidelberg (2005)
17. PASSI: Toolkit web page, http://sourceforge.net/projects/ptk
18. TAOM4E: Home page, http://sra.itc.it/tools/taom4e/
19. Garijo, F.J., Gòmez-Sanz, J.J., Massonet, P.: The MESSAGE methodoly for agent-oriented analysis and design. In: [43], ch. VIII, pp. 203–235
20. Caire, G., Coulier, W., Garijo, F.J., Gomez, J., Pavòn, J., Leal, F., Chainho, P., Kearney, P.E., Stark, J., Evans, R., Massonet, P.: Agent oriented analysis using Message/UML. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 119–135. Springer, Heidelberg (2002)
21. Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: [43], ch. IX, pp. 236–276
22. JADE: Home page (2000), http://sharon.cselt.it/projects/jade/
23. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a fipa-compliant agent framework. Softw., Pract. Exper. 31, 103–128 (2001)
24. Bernon, C., Cossentino, M., Gleizes, M.P., Turci, P., Zambonelli, F.: A study of some multi-agent meta-models. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 62–77. Springer, Heidelberg (2005)
25. van Hillegersberg, J., Kumar, K., Welke, R.J.: Using metamodeling to analyze the fit of object-oriented methods to languages. In: 31st Hawaii Inter. Conference on System Sciences (HICSS 1998), Modeling Technologies and Intelligent Systems, Kohala Coast, HI, USA, vol. 5, pp. 323–332. IEEE Computer Society Press, Los Alamitos (1998)
26. Molesini, A., Omicini, A., Ricci, A., Denti, E.: Zooming multi-agent systems. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, pp. 81–93. Springer, Heidelberg (2006)
27. Molesini, A., Omicini, A., Denti, E., Ricci, A.: SODA: A roadmap to artefacts. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, pp. 49–62. Springer, Heidelberg (2006)
28. Omicini, A., Ricci, A., Viroli, M.: Coordination artifacts as first-class abstractions for MAS engineering: State of the research. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 71–90. Springer, Heidelberg (2006)
29. Omicini, A., Ricci, A., Viroli, M.: Agens Faber: Toward a theory of artefacts for MAS. In: Jacquet, J.-M., Picco, G.P. (eds.) COORDINATION 2005. Electronic Notes in Theoretical Computer Sciences, vol. 150, pp. 21–36 (2005)
30. Omicini, A.: Formal ReSpecT in the A&A perspective. Electronic Notes in Theoretical Computer Sciences 175, 97–117 (2007); Post-proceedings of 5th Inter. Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA 2006), CONCUR 2006, Bonn, Germany August 31 (2006)
31. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems 7, 80–112 (1985)
32. Gelernter, D., Carriero, N.: Coordination languages and their significance. Communications of the ACM 35, 97–107 (1992)

33. Omicini, A.: Towards a notion of agent coordination context. In: Marinescu, D.C., Lee, C. (eds.) Process Coordination and Ubiquitous Computing, pp. 187–200. CRC Press, Boca Raton (2002)
34. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: An infrastructure for engineering computational environments in MAS. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006, pp. 102–119 (2006)
35. Parunak, H.V.D.: Go to the ant: Engineering principles from natural agent systems. Annals of Operation Research 75, 69–101 (1997)
36. Viroli, M., Omicini, A., Ricci, A.: Engineering MAS environment with artifacts. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2005, pp. 62–77 (2006)
37. Papadopoulos, G.A., Arbab, F.: Coordination models and languages. Advances in Computers 46, 330–401 (1998)
38. OMG: Home page, http://www.omg.org/mda/
39. Gracanin, D., Singh, H.L., Bohner, S.A., Hinchey, M.G.: Model-driven architecture for agent-based systems. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 249–261. Springer, Heidelberg (2004)
40. Amor, M., Fuentes, L., Vallecillo, A.: Bridging the gap between agent-oriented design and implementation using MDA. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 93–108. Springer, Heidelberg (2005)
41. Cabri, G., Leonardi, L., Zambonelli, F.: MARS: A programmable coordination architecture for mobile agents. IEEE Internet Computing 4(4), 26–35 (2000)
42. Sycara, K.P., Paolucci, M., Velsen, M.V., Giampapa, J.A.: The RETSINA MAS infrastructure. Autonomous Agents and Multi-Agent Systems 7, 29–48 (2003)
43. Henderson-Sellers, B., Giorgini, P. (eds.): Agent Oriented Methodologies. Idea Group Publishing, Hershey (2005)

Model Driven Engineering for Designing Adaptive Multi-Agents Systems

Sylvain Rougemaille, Frédéric Migeon, Christine Maurel, and Marie-Pierre Gleizes

IRIT – Paul Sabatier University – 118, Route de Narbonne
31062 Toulouse, Cedex 9, France
Tel.: +33 561 558456
{rougemaille,migeon,maurel,gleizes}@irit.fr

Abstract. A challenge for our days is to provide new efficient CASE (Computer Aided Software Engineering) tools enabling MAS designers towards Model Driven Engineering (MDE) approaches. The goal of MDE is to improve the development process and the quality of the software produced. Our work focuses on two different aspects of MAS. The functional one, which is application dependent and close to the decision process of agents, and the operational one related to elementary capabilities of agents. For each point of view, we have defined specific meta-models. Our goal in this paper is to provide a mapping from the functional meta-model to the operational that constitutes a specific platform model. As we are interested in adaptive systems, we have to deal with adaptation both at the agent and the system level. We address this problem by respectively using the JavAct flexible architecture and the Adaptive MAS principles.

1 Introduction

A challenge for our days is to provide new efficient CASE (Computer Aided Software Engineering) tools enabling MAS designers towards Model Driven Engineering (MDE) approaches. The goal of MDE is to improve the development process and the quality of the software produced. Basically, Model Driven Architecture (MDA) [1] proposes to automatically generate the PSM (Platform Specific Model) by merging early specifications expressed as the PIM (Platform Independent Model) and some intermediary PM (Platform Model) by using several automated transformations.

The work presented in this paper consists of the enhancement of the existing ADELFE methodology [2], based on the AMAS (Adaptive Multi-Agent Systems) theory and dedicated to the design of self-organising systems. The aim is to enrich ADELFE with a development phase providing tools based on MDE. In particular, we want to focus on two aspects of a MAS: the functional aspect which is application dependent and close to the decision process of agents, and the operational related to elementary skills of agents. For each point of view, we have defined specific meta-models, we support mapping from the functional meta-model to the operational one avoiding the merging phase of a classical MDA approach.

Furthermore, systems we are interested in must be adaptive and we take into account the adaptation at both agent and system levels. We have developed the JavAct

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 318–332, 2008.

agent-based middleware [3] whose agents are designed to be adaptive by the means of a component-based flexible architecture and which is a particularly well suited target for implementing an AMAS whose main principle is system self-adaptation. However, JavAct is used as an implementation platform for our work and is not in the scope of this paper. We simply use its adaptation capabilities thanks to its architecture and promote it at the model level.

In this paper, we advocate that the design of an agent must be realized by considering two different levels: an operational one and a functional one (Section 2). Once the proposed approach is positioned in relation to existing works (Section 3), different points of view of the meta-models, at the two levels previously determined, are described (Section 4). Then, the mapping enabling the transformation from a functional meta-model to an operational one is defined (Section 5). The paper ends with a discussion and a presentation of the perspectives of the proposed approach (Section 6).

2 Operational and Functional Adaptation

In living systems, adaptation is linked to the species' learning and evolution capabilities according to the Darwin's theory [4]. Thus, adaptation is a process enabling changing systems' structure and\or behaviour in order for the species to survive. In artificial systems, adaptation can occur as an off-line manner, for example when a programmer stops the code execution, changes a line of code and launches it again. But, the more complex and interesting concept of adaptation which is treated in this paper, is self-adaptation in which the artificial system performs some internal changes, during the execution to enable the system adaptation. The reason why an artificial system has to self-adapt is explained in [5]. According to the DARPA definition, Robertson and al. said: "self-adaptive software evaluates its own behaviour and changes behaviour when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible".

The systems that we are focusing on can be an agent or a MAS. So, to self-adapt an agent or a MAS has to change; i.e. to modify its behaviour or a part of its structure. For example, an artificial ant can learn that when it is in front of a wall, it has to turn (behaviour) or if it possesses the adequate skills, it can self-add one leg (adding a part). We can assume that, in general, the adaptation process starts by an interaction between the system and its environment. As Maturana and Varela have shown, there exists a coupling between the system and the environment [6], there exists a very close relation between a system and the environment in which it is executed. This relation is based on the system perceptions of its environment and the actions it makes to deal with. Once deployed, the system has to react to its surrounding environment stimuli, one way to react is, in particular, to self-adapt.

The designer must lead an analysis at the system (or global level), and at the agent (or local level) in order to design a self-adaptive MAS. Most of the works on multi-agent design agree that an agent follows the life cycle composed of three main steps: perceive, decide and act. In each step, the agent design must take into account two levels: an operational one and a functional one.

Considering the perceive-decide-act lifecycle, we understand those levels as follow. The operational level is made up of all the tools or means which enable the perception

of the environment and the performance of actions such as moving, sending messages. The functional level is related to all the means which enable the choice of an action to perform such as "if there is an obstacle in front of me I have to turn left or right", i.e. the decision process. Those levels are detailed in the next sections.

2.1 Operational Point of View

The operational part of an application gathers everything which is independent of the application layer. Therefore, operational adaptation can be seen as updating the system to its execution environment, without altering its behaviour, so that it is more efficient or simply compliant with it. For example, an agent entering an unsecured zone could have its message sending protocol become encrypted, independently of its functional behaviour.

For this purpose, we developed the JavAct middleware [3] where agents benefit from a meta-level architecture. While the base-level contains the functional part of an agent, the meta-level gives flexibility to this agent by means of interconnected micro-components; each of them implements an operational skill of the agent (mobility, message sending, environment perception, dynamic creation etc.). Consequently, operational adaptation is obtained by changing a micro-component dynamically. Of course, operational adaptation can be triggered by functional concerns and is defined by the programmer at design time, i.e. concerns and conditions from the base level imply meta-level modifications. For example, the encrypted communication could be a small part of a complex and functional adaptation of an agent which perceives its environment as unsecured and decides that encryption is the best way to adapt.

2.2 Functional Point of View

The functional behaviour can be viewed as the result produced by a system in an applicative context. It is what the system can realize to achieve the requirements of an application and usually is domain dependent. MAS adaptation can be due to change either in individual behaviour of agents or in the collective behaviour of the system. In this paper, we focus on functional adaptation at the system level.

At the agent level, the function is related to the result provided by the agent action and self-adaptation consists in changing the decision process leading to another action, or learning a new action to be done. The adaptation of its behaviour can be realized by endowing the agent with a learning process. With this ability, the agent can act differently and can adapt its behaviour to subsequent events in its environment by using the knowledge learnt from making previous decisions. The research community on agent learning has produced a lot of work such as case-based reasoning, reinforcement learning, neural networks, etc. which are usually used in the MAS community.

System adaptation consists in changing the behaviour of the collective in response to new or modified environments. In the AMAS (Adaptive Multi-Agent Systems) approach [7], we have worked to highlight a generic behaviour which enables the system to self-adapt. The AMAS theory provides a guide to design self-organising systems. It is based on the observation that modifying interactions between the agents of the system modifies also the global function and makes the system self-adapt to changes in its environment. The local criterion used by agents to decide which kind of

action they perform, is the cooperative attitude. An agent behaviour is led by the two following attitudes: a repairing one and an anticipative one. If no change appears in its environment, an agent performs its nominal behaviour and if changes occur it analyses the situation and chooses the most cooperative action related to its environment in which other agents evolve.

According to the AMAS theory, every agent pursues an individual objective and interacts with agents it knows by respecting cooperative techniques which lead to the avoidance of Non Cooperative Situations (NCS) like conflict or concurrence. Faced with a NCS, a cooperative agent acts to come back to a cooperative state and permanently adapts itself to unpredicted situations while learning from others. The AMAS theory is based on how an agent can avoid failures and this approach is an exceptions handling mechanism at the agent level.

2.3 Adaptation Levels

As we have seen in this section, we have identified different kinds of adaptation but also different levels of concerns. The following table sums up the idea:

	Functional Adaptation	**Operational Adaptation**
Agent	Classical learning approaches	JavAct
System	AMAS approach	

Operational adaptation in JavAct mainly concerns the agent itself whereas the AMAS approach fits functional adaptation of the system. However, it is easy to obtain operational adaptation of the system by coordinating operational adaptation of agents as well as functional adaptation of agents (such as learning for example), which can be obtained from AMAS concerns. The operational mechanisms involved in the execution of an AMAS are at least, constituted by the set of the mechanisms used by each type of agent. The complexity of the operational adaptation mechanism of the whole system would depend on the heterogeneity of the agents of which it is composed. Each agent is responsible of its own operational adaptation, thus the system operational adaptation only depends on its agents. Furthermore, the consistency should be guaranteed by cooperation rules that insure for example the understanding of messages.

So, our aim is the providing of means to design self-adaptive MAS which enable us to take into account in one tool the operational adaptation at the agent level and functional adaptation at the system level. We have developed the ADELFE method [8], to design AMAS but this method consists only of the three first steps of the design life cycle: Requirements, Analysis and Design Workflows. Our aim is to add the development phase to ADELFE by taking into account the two previous levels of adaptation. Because the objective is to reduce the design duration and the complexity of the task for designers, they should only focus on the system functional adaptation (NCS definition) and the agent definition while the operational adaptation should be automatically handled by the JavAct middleware thanks to model transformations. Mapping AMAS operational concerns to JavAct specific architecture (see section 5) is the purpose of the presented work and is the first step to reach that goal.

3 Related Work

Currently, most of the existing agent-based methodologies [9], [10] have fully taken into account the first phases of a software development life cycle: requirements, analysis and design phases. The phases such as implementation, test, deployment and maintenance are more or less treated in the following well-known methods ADELFE [8], INGENIAS [11], PASSI [12], and TROPOS [13]. But an effort has been made on these phases, notably in proposing new tools to facilitate code generation. In this section, the main works using models transformation in order to design MAS are reviewed and the main tools coming from MDE are analysed.

3.1 MAS Related Work

The first result to improve the agent-based software development is about meta-models definition in MAS methods and the second is about the use of model transformations coming from the MDA or MDE community in order to generate automatically the code for a given platform.

3.1.1 MAS Meta-models

Since 2003, initiated by FIPA (Foundation for Intelligent Physical Agents) working groups, several meta-models have been defined. The difficulty was to find a unique and agreed meta-model, so several meta-models have been defined.

AALAADIN [14] is a meta-model based on agent, group and role concepts. It enables principally the description of organisation; agents belong to groups in which they handle roles. Agent are intentionally not detailed, thus, developers are free to choose the one that better fit their requirements. A concrete adoption of AALAADIN is used in the MadKit platform [14] in the requirements and design phases.

FAML [15] meta-model is a meta-model built to take into account every kind of existing MAS. It is expressed in two layers: design-time and runtime, concerning both the agent and the system point of views. At design level, the expressed concepts are role, task, agent, plan, action, ontology and environment access points. At runtime, environment, events, system access points, plans, action, message, desires, beliefs and intention. At each level, the relationships between concepts are explained.

GAIA [16] meta-model highlights the notions of: role (which is refined in responsibility, activity, permission concepts), agent, communication (with protocols), organisation (structure and rules) and environment (resource).

INGENIAS [11] meta-model integrates different results on multi-agent and agent works. By consequence, it considers a MAS from five complementary viewpoints:

- Organization (workflow, group, agents, roles, resources, and applications),
- Agent (tasks, goals, mental states and roles),
- Tasks/goals which describes their decomposition and the consequences of their execution in terms of: mental entity, interaction, resource, application,
- Interactions (agent, goal, role, task, specification),
- And environment (agent, application, resource).

PASSI meta-model [12] is composed of three domains: the solution, the agency and the problem domains. The problem domain deals with the user's problem in terms of scenarios, requirements, ontology and resources. In the agency domain, the main concepts are agent, role, task and communication with AIP message. The implementation domain describes the structure of the code solution in the chosen FIPA-compliant implementation platforms.

TROPOS [13] is a method organized in five phases: early requirements, late requirements, architectural design, detailed design and implementation. The main concepts enabling expression of intentional and social concepts are: actor, goal, plan, resource and the relationships between them such as actor dependency goal decomposition, plan decomposition, means-end and contribution relationships.

Some attempts have been made such as the gathering of ADELFE, GAIA and PASSI meta-models [17] but the meta-model obtained was seen as being to complex so that the authors didn't pursue this work. It seems better to define different meta-models in relation with the type of MAS to be designed. However, these works on meta-models have given us a better understanding of the concepts used in the MAS community and lead us to the use of MDA or MDE tools.

3.1.2 MAS and MDA/ MDE

Few works on MAS engineering have integrated tools coming from MDE, and the most advanced are: INGENIAS[1], MetaDIMA [18] and TROPOS [19].

MetaDIMA helps the designer to implement MAS on the DIMA platform using MetaGen which is a MDE tool dedicated to the definition of meta-models and models. DIMA is a development and implementation platform developed in Java where agents are seen as a set of dedicated modules (perception, communication etc.). MetaDIMA provides a set of meta-models and a set of knowledge-based systems on top of DIMA to ease the design of MAS by providing languages more specific than Java code.

In TROPOS (see 0), all the phases use different models which are described by meta-models; it also uses UML notation and automatic transformations. For example, it translates plan decomposition into a UML 2.0 activity diagram by using a transformation language based on the following three concepts: pattern definition, transformation rules and tracking relationships.

INGENIAS proposes to transform the MAS expressed in the INGENIAS meta-model in code dedicated to a given platform using the modelling language of INGENIAS and the implementation model of the platform. Its main originality consists in providing evolutionary tools. Because tools used for transforming specification in code are based on meta-models, if the meta-model specifications evolve the tools can also evolve.

Our work pursues the same objective as the works described previously although it addresses the adaptation issue from both system and agent points of view. In fact, we aim at taking it into account and providing design and generation tools to implement such adaptive systems. For this purpose, we propose to generate an adapted execution platform for AMAS, using MDE tools and principles as well as the flexibility of the JavAct middleware.

[1] http://grasia.fdi.ucm.es/ingenias/

3.2 MDE Tools

This section presents a brief overview of tools we have already used or foresee using in the model driven scope.

3.2.1 Model Editing Tools

The eclipse IDE provides the *EMF* (Eclipse Modeling Framework) [20] which provides a meta-modelling language called *Ecore* (allowing description of meta-models) and features to edit, handle and modify models. On top of this plug-in, we use *Topcased* [21] as an *Ecore* editor and a graphical editor generator, i.e. we define editors for our different modelling languages thanks to its generative capabilities. With the same purpose, we have compared *Topcased* to *GMF* (Graphical Modeling Framework) which possesses more or less the same functionalities while adopting a different approach for graphical editor description where each aspect of the editor is described by a model.

3.2.2 Transformation Tools

We plan to use model to model transformations to implement our mapping (see section 5.) and model to text transformations to generate JavAct code. Those transformations have to be supported by tools and languages. We mainly focus on *ATL* (Atlas Transformation Language) [22] and *Kermeta* [23] which both provide tools based on *EMF* and are implemented as Eclipse plug-in. *ATL* is a hybrid language providing declarative features while *Kermeta* is defined as a meta-programming language close to OO programming languages. However, we plan to use their specificities respectively to implement transformations and to equip our modelling language with execution capabilities (for simulation purpose).

4 Meta-models

The main idea of our work can be summed up as follow. On the one hand we have the AMAS theory which intrinsically implies to deal with self-adaptation and on the other one, we have the JavAct middleware whose agents are designed to be flexible and adaptive (on the operational point of view). Our purpose is to bridge the gap between them as automatically as possible. To achieve this task we assume to tackle this problem at the highest abstraction level, i.e. at meta-model level, and use model transformations to build that bridge.

To describe both JavAct agent model and AMAS in a model driven approach, we have developed two dedicated modelling languages (DSML[2]) which are themselves described by meta-models. Those meta-models can be seen as representations of the main concepts and relationships we have identified for each of these particular domains (AMAS and JavAct micro-architecture). Our idea is to map automatically agents from an AMAS model to an adapted JavAct micro-Architecture model; this could be done using model transformation languages as presented in the previous section. Thus, it is necessary to describe as precisely as possible what are the key concepts of the two domains. The following sections give a brief overview of those meta-models.

[2] Domain Specific Modeling Language.

4.1 AMAS Meta-model

The AMAS meta-model was elaborated from the ADELFE meta-model [17] enriched by three distinct logical points of view to describe an AMAS. Each of them represents a specific part of the AMAS theory on which we want to put emphasis:

- *System point of view*: it is devoted to the description of the system and its surroundings in terms of *Entities* which populate it (perceptible objects of the "world").
- *Agent point of view*: this part of the meta-model represents agent internal characteristics.
- *Cooperation point of view*: it represents taxonomy of *Non-Cooperative Situations* an AMAS agent is likely to encounter.

In the context of this paper, we focus on the agent point of view.

4.1.1 AMAS Agent Point of View

An AMAS agent is made up of various modules, parts, managing a sector of its activities and life-cycle. Typically, the AMAS agent life-cycle is defined according to the three phases: *perception*, *decision* and *action*. From these phases and the needs they imply in terms of environmental interactions, knowledge representation, non-cooperative situations avoidance, etc. we determine the following meta-model concepts:

- Environmental interactions are represented by *Perception* and *Action* on the *Entities* as well as the means to carry them out (*Actuator*, *Sensor*). *CommunicationAction* consists of direct interaction with other agents by the means of messages (*Message*) whose protocol is defined in the *System* point of view (*CommunicationProtocol*).
- *DecisionModule* gathers the *Aptitudes* and the *CooperationRules* enabling an agent to determine the next actions to lead. This decision is taken according to agent knowledge and aptitudes as well as its cooperation rules which propose actions to overcome possibly detected *NonCooperativeSituations (NCS).* Without NCS detection, an agent carries out its local and nominal function determined by its aptitudes.
- *Knowledge* represents what an agent possesses. It has self-*Representations* and *Representations* of the medium surrounding it (Environment *Entities,* agents of the system). It also possesses *Skills* and its *Characteristics* possibly perceptible by other agents of the system (*isPerceptible*).

4.2 Micro-architecture Description Language (µADL) Meta-Model

µADL is a micro-architecture description language based on the previously expounded principle of JavAct micro-architecture [6] (cf. section 2). It focuses on operational mechanisms definition of JavAct agents in terms of micro-components. A new micro-component assembly is called a micro-architecture and it constitutes a new agent "style" which can be used in a JavAct application. Thus, µADL provides the concepts of *MuArchitecture*, *MuComponent* and of *Interface* (see Figure 1). A *MuArchitecture*

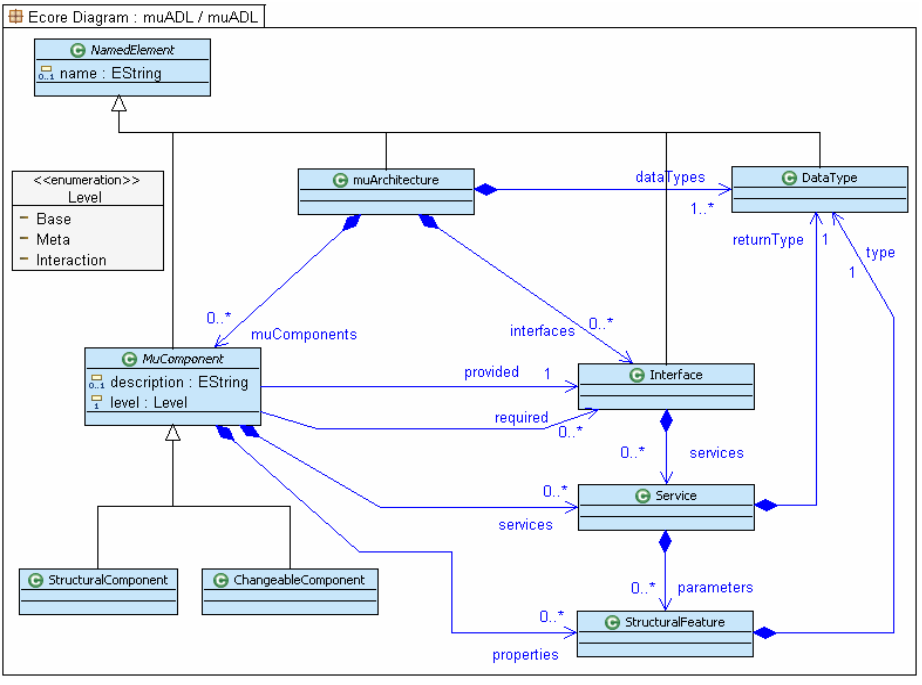


Fig. 1. Micro-Architecture meta-model

is a composition of *MuComponents* connected to each other by the fact that they provide or require *Interfaces*. We have defined two different *MuComponents*:

- *StructuralComponents*: corresponding to the micro-components that are not intended to be redefined or to be substituted. According to the policy defined in JavAct, they represent core concepts of the architecture that are used to achieve reflection and delegation mechanisms.
- *ChangeableComponents*: conversely, they correspond to the micro-components which could be replaced, modified, extended and so on.

Each JavAct micro-component is defined by the following fields:

- *name*: its name,
- *description*: a short informal description of its operating properties,
- *level*: the level in which it offers its services (*base, meta, interaction*),
- *properties*: data that could be necessary to achieve a particular service,
- *services:* the *Services* it implements.
- *provided*, *required*: the *Services* it requires and it offers to other *MuComponents* through *Interfaces*.

StructuralFeatures and *DataType* are both meta-classes that enable the definitions of *MuComponent* properties which can be assimilated to typed attibutes in the Object world.

JavAct micro-architectures are forced by the fact that they always have to delegate services provided by the base level *μComponents* to the agent functional code. It implies that the micro-architectures contain at least one *Controller* and one *meta-access* component. This kind of constraint is verified with dedicated OCL rules.

5 The Mapping Process (Mapping AMAS Agents to the JavAct Platform)

To instantiate a particular JavAct agent model for each agent within the AMAS model, we have to map concepts from the AMAS meta-model to μADL ones. At this time, we simply focus on environmental interactions of AMAS agent (i.e. Action, Perception, etc.). Furthermore, we present an "informal" mapping that could be implemented with model transformation languages.

5.1 Meta-models Mapping

The mapping takes place between the AMAS and the μADL meta-models, so we have to specify which concepts have to be mapped to each others. Thus, the aim of this mapping is to operationally adapt JavAct agent so that they become compatible with the AMAS theory, that is to say: bring them to a higher degree of environmental interaction and cooperation. This can be done by expressing intrinsic characteristics of AMAS agents in terms of *MuComponents*. In other words, we focus on the operational aspects of AMAS agents and we try to describe them as a micro-architecture. To express more conveniently this mapping, we provide an overview table which represents μADL concepts for each AMAS meta-class (see Table 1).

Each *muComponent* has to provide at least one *Interface* containing at least one *Service* of those implemented by the *muComponent*. This is a generic mapping rule: we consider that *Service* providing is necessarily done through those *muComponent* related interfaces.

Most of the AMAS meta-classes map to *MuComponents*, although there are exceptions to this rule. Those exceptions are related to particular implementation choices we made. To illustrate those choices, consider the *Knowledge* meta-class and subclasses. Knowledge is a particularly important part of a cooperative agent because of its implication in the decision process; thus, we decide to reify this concept in term of a *MuComponent*. This *MuComponent* is related to two more specific *Skill* and *Characteristic MuComponents*, which are designed to embody respectively all skills and characteristics of cooperative agents as vectors.

Another important point of interest is the *MuComponent* level, which specifies the scope of a *MuComponent*, i.e. whether it can be accessed by the functional code (*base* level) or not (*meta* level). The *Interaction* level represents *MuComponents* dedicated to agent external interactions. For example, the *Action perform()* service is used to define reaction to *NCS*, which is the purpose of AMAS agent functional code; thus we define *Action* as a *base* level *MuComponent*.

As we define a meta-level mapping between AMAS and μADL meta-models, the next section presents what should be the result of a transformation at model level. This is done using a simple and well-known AMAS example and focusing on environmental interactions.

Table 1. Mapping AMAS meta-classes to µADL concepts

AMAS		µADL					
		MuComponent					
Meta-class name		Name	Level	Services	Provided	Required	Properties
Action	Maps to	Actions	base	performAction()	performAction()	perform()	-
							actionList
Communication Action		Communication	base	send() receive()	send() receive()	-	protocol
Message							messageType
Perception		Perception	meta	perceive()	perceive()	perceive()	-
Actuator		Actuator	interaction	Enabled actions	Enabled actions	-	-
Sensor		Sensor	interaction	Enabled perceptions	Enabled perceptions	-	-
Knowledge		Knowledge	base	update() getRepresentation()	update() getRepresentation()	update() getRepresentation()	
Representation							repList (element)
Skill		Skills	meta	getAction() update() getRepresentation()	getAction() update() getRepresentation()		
							skillList (element)
Characteristic		Characteristics	meta	update() getRepresentation()	update() getRepresentation()		
							characteristicList (element)
Decision Module		Decision	meta	decide()	decide()	getRepresentation() getAction()	
Agent		LifeCycle	base	run()	run()	perceive() decide() perform()	

5.2 Model Mapping: Ants Example

This example comes from the ANTS project [24], whose purpose was to define software ant-robots based on ethological observations concerning the foraging process of an anthill. The aim was not to simulate the real process but use the available information

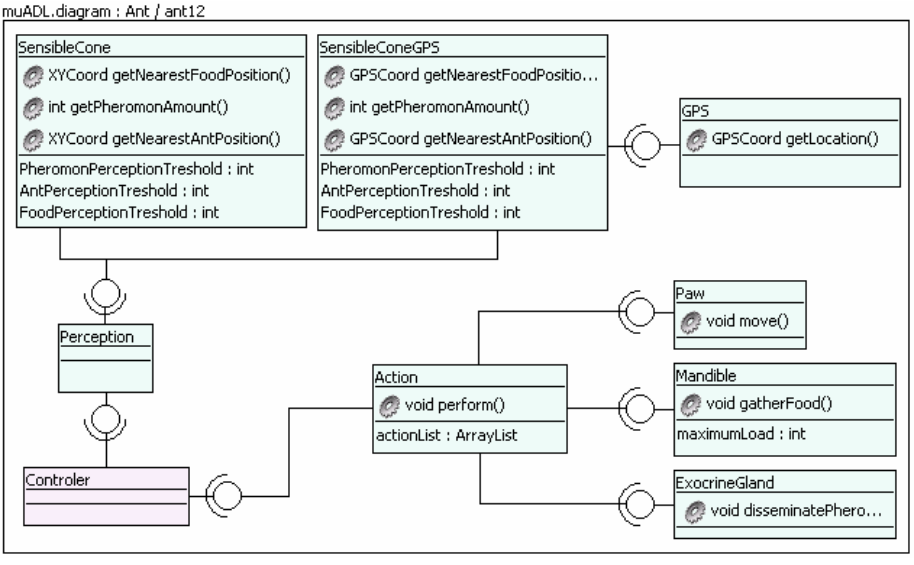


Fig. 2. Part of an ant agent µADL graphical model (informal mapping result)

coming from the biologists to implement robots which have to collect distributed resources in an unknown environment.

In this part, we focus on the environmental interactions described with the AMAS meta-model and what should be the result of the mapping in the µADL meta-model. In other words, the µADL model specifies the appropriate operational mechanisms of a JavAct cooperative ant-agent.

From an AMAS point of view, an ant agent possesses a *SensibleCone* which allows it to perceive food, pheromones and other ants of the colony. This sensor is qualified by properties limiting its scope (threshold area values for food, pheromones and ants). Ants are also able to explore their environment (thanks to *move action*), to gather food when they find some and if the load is not too heavy (*gatherFood action*) and to disseminate pheromones on their way back to the nest (*disseminatePheromon action*). All those actions are carried out by actuators. By focusing on this part of the AMAS ant colony model and applying our mapping we obtain the following µADL model (see Figure 2), in which all actuators have been mapped to micro-components (*Paw*, *Mandible* and *ExocrineGland*) as sensor has been (*SensibleCone*).

JavAct micro-architecture enables dynamic operational adaptation by switching two micro-components provided that they implement the same interface. Figure 2 presents a static vision of that capacity (µADL model). At runtime it is possible for an ant JavAct agent to choose the more appropriate way to "sense" its environment. For instance, a robot ant evolving on a real tangible terrain should use the *SensibleConeGPS* micro-component in order to locate itself and its surroundings. But if the GPS device is damaged JavAct enables the ant agent to switch the useless component with the *SensibleCone* micro-component which uses a simulated environment.

6 Conclusion and Perspectives

In this paper, we present our work whose aim is to add the development phase to ADELFE by considering two adaptation levels: a functional one for the system, and an operational one for JavAct agents. As the goal is to reduce the duration and the complexity of the design of AMAS, we investigated a MDE approach; this is based on model transformations in order to facilitate code production for a given platform.

As we aim to provide JavAct agent version which be automatically fitted to the Adaptive Multi-Agent System (AMAS), we have developed two dedicated modelling languages (DSML) with meta-models which describe respectively JavAct agent and AMAS.

As we have seen, adaptation is the central point of our methodology and tools. The designer will be able to describe functional adaptation in models which conform to the AMAS meta-model. The mapping described in the previous section will then result in a model of a specific JavAct agent architecture. This model can be considered as a Platform description Model (PM) adapted to the nature of the application. Therefore, the gain is two-fold. On the one hand, we combine functional adaptation of AMAS and operational adaptation of JavAct agents. On the other one, we simplify the complexity of design: details of implementation are hidden during early conception phases. In continuation of this work, we are studying the different stages for implementing a concrete prototype of the CASE tool. As was mentioned in section 3.2, we have already used Topcased or GMF tools to generate automatically graphical editors for the DSML defined. We hope now to be able to automate the production of these editors from meta-models such as the AMAS one. Of course, this cannot be done from the AMAS meta-model only. Some additional information must be collected in order to automate to the generation of the editors.

Finally we have also some experiences in the production of JavAct code using an ad-hoc generator we developed. This tool, called Agent$^{\varphi}$, allows generation of JavAct code from an assembly description. However, work has still to be done for generalizing its use and for integrating it in a MDE tool.

Acknowledgments

We would like to thank Carole Bernon, Thierry Millan and Pierre Glize for discussions about the meta-models.

References

[1] OMG, MDA Guide, Object Management Group, Inc., Final Adopted Specification (2003)
[2] Bernon, C., Gleizes, M.-P., Peyruqueou, S., Picard, G.: ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering. In: Petta, P., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2002. LNCS (LNAI), vol. 2577. Springer, Heidelberg (2003)
[3] Leriche, S., Arcangeli, J.P.: Adaptive Autonomous Agent Models for Open Distributed Systems. In: International Multi-Conference on Computing in the Global Information Technology (ICCGI 2007), March 2007, pp. 19–24. IEEE Computer Society, Los Alamitos (2007)

[4] Darwin, C.: On the Origin of Species by Means of Natural Selection. John Murray, London (1859)
[5] Robertson, P., Laddaga, R., Shrobe, H.: Introduction: the First International Workshop on Self-Adaptive Software. In: Robertson, P., Shrobe, H.E., Laddaga, R. (eds.) IWSAS 2000. LNCS, vol. 1936, pp. 1–10. Springer, Heidelberg (2001)
[6] Varela, F., Maturana, H.: The Tree of Knowledge: The Biological Roots of Human Understanding. Shambhala Press, Boston (1998)
[7] Capera, D., Georgé, J.-P., Gleizes, M.-P., Glize, P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In: Proc. 12th IEEE International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises, Linz, Austria, June 9-11, pp. 383–388. IEEE Computer Society, Los Alamitos (2003)
[8] Bernon, C., Gleizes, M.-P., Picard, G.: Enhancing Self-Organising Emergent Systems Design with Simulation. In: International Workshop on Engineering Societies in the Agents World (ESAW 2006), Dublin (September 2006)
[9] Bergenti, F., Gleizes, M.-P., Zambonelli, F. (eds.): Methodologies and Software Engineering for Agent Systems. Kluwer, Dordrecht (2004)
[10] Henderson-Sellers, B., Giorgini, P. (eds.): – Agent-Oriented Methodologies. Idea Group Pub. (June 2005)
[11] Gomez Sanz, J., Fuentes, R.: Agent Oriented System Engineering with INGENIAS. In: Fourth Iberoamerican Workshop on Multi-Agent Systems, Iberagents 2002 (2002)
[12] Cossentino, M.: From Requirements to Code with the PASSI Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, June 2005, pp. 79–106. Idea Group Pub. (2005)
[13] Giorgini, P., Kolp, M., Mylopoulos, J., Castro, J.: Tropos: A Requirements-Driven Methodology for Agent-Oriented Software. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent Oriented Methodologies, pp. 20–45. Idea Group (2005)
[14] Gutknecht, O., Michel, F., Ferber, J.: The MadKit Agent Platform Architecture, Research Report, LIRMM (April 2000)
[15] Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B., Low, G.: Developing and Evaluating a Generic Metamodel for MAS Work Products. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 126–142. Springer, Heidelberg (2006)
[16] Cernuzzi, L., Juan, T., Sterling, L., Zambonelli, F.: The Gaia Methodology: Basic Concepts and Extensions. In: Bergenti, F., Gleizes, M.-P., Zambonelli, F. (eds.) Methodologies and Software Engineering for Agent Systems. Kluwer Academic Publishers, Dordrecht (2004)
[17] Bernon, C., Cossentino, M., Gleizes., M.-P., Turci, P., Zambonelli, F.: A study of some Multi-Agent Meta-Models. In: Giorgini, P., Mueller, J.P., Odell, J. (eds.) The Fifth International Workshop on Agent-Oriented Software Engineering (AOSE 2004), New York, USA, July 19 (2004)
[18] Guessoum., Z., Jarraya, T.: Meta-Models & Model-Driven Architectures, Contribution to the AOSE TFG AgentLink3 meeting, Ljubljana (2005)
[19] Perini, A., Susi, A.: Automating Model Transformations in Agent-Oriented Modelling. In: Proceedings of 6th International Workshop AOSE 2005, Utrecht, NL, July 25-26 (2005)
[20] Budinsky, F., Steinberg, D., Ellersick, R.: Eclipse Modeling Framework: A Developer's Guide. Addison-Wesley Professional, Reading (2003)

[21] Farail, P., Gaufillet, P., Canals, A., Camus, C.L., Sciamma, D., Michel, P., Crégut, X., Pantel, M.: TOPCASED project: a Toolkit in OPen source for Critical Aeronautic SystEms Design. In: Embedded Real Time Software (ERTS) (2006)
[22] Jouault, F., Kurtev, I.: Transforming Models with ATL. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaic (2005)
[23] Muller, P., Fleurey, F., Jézéquel, J.: Weaving Executability into Object-Oriented Meta-Languages. LNCS, Montego Bay, Jamaica. Springer, Heidelberg (2005)
[24] Topin, X., Fourcassié, V., Gleizes, M.-P., Théraulaz, G., Régis, C., Glize, P.: Theories and experiments on emergent behaviour: From natural to artificial systems and back. In: Proceedings on European Conference on Cognitive Science, Siena (1999)

Trace-Based Specification of Law and Guidance Policies for Multi-Agent Systems*

Scott J. Harmon, Scott A. DeLoach, and Robby

Kansas State University, Manhattan KS 66506, USA
{harmon,sdeloach,robby}@ksu.edu

Abstract. Policies have traditionally been a way to specify properties of a system. In this paper, we show how policies can be applied to the Organization Model for Adaptive Computational Systems (OMACS). In OMACS, policies may constrain assignments of agents to roles, the structure of the goal model for the organization, or how an agent may play a particular role. In this paper, we focus on policies limiting system traces; this is done to leverage the work already done for specification and verification of properties in concurrent programs. We show how traditional policies can be characterized as *law policies*; that is, they must always be followed by a system. In the context of multiagent systems, law policies limit the flexibility of the system. Thus, in order to preserve the system flexibility while still being able to guide the system into preferring certain behaviors, we introduce the concept of *guidance policies*. These *guidance policies* need not always be followed; when the system cannot continue with the *guidance policies*, they may be suspended. We show how this can guide how the system achieves the top-level goal while not decreasing flexibility of the system. *Guidance policies* are formally defined and, since multiple *guidance policies* can introduce conflicts, a strategy for resolving conflicts is given.

1 Introduction

As computer systems have been charged with solving problems of greater complexity, the need for distributed, intelligent systems has increased. As a result, there has been a focus on creating systems based on interacting autonomous agents. This investigation has created an interest in multiagent systems and multiagent system engineering, which proscribes formalisms and methods to help software engineers design multiagent systems. One aspect of multiagent systems that is receiving considerable attention is the area of policies. These policies have been used to describe the properties of a multiagent system–whether that be behavior or some other design constraints. Policies are essential in designing societies of agents that are both predictable and reliable [1]. Policies have traditionally been interpreted as properties that must always hold. However, this does not capture the notion of policies in human organizations, as they are often used as normative *guidance*, not strict *laws*. Typically, when a policy cannot be followed in a multiagent system, the system cannot achieve its goals, and thus, it cannot continue

* This work was supported by grants from the US National Science Foundation (0347545) and the US Air Force Office of Scientific Research (FA9550-06-1-0058).

A. Artikis et al. (Eds.): ESAW 2007, LNAI 4995, pp. 333–349, 2008.

to perform. In contrast, policies in human organizations are often suspended in order to achieve the overall goals of the organization. We believe that such an approach could be extremely beneficial to multiagent systems residing in a dynamic environment. Thus, we want to enable developers to guide the system without constraining it to the point where it cannot function effectively or looses its autonomy.

The main contributions of this paper are: (1) a *formal* trace-based foundation for *law* (must always be followed) and *guidance* (need not always be followed) policies, (2) a conflict resolution strategy for choosing between which guidance policies to violate, and (3) validation of our approach through a set of simulated multiagent systems.

The rest of the paper is organized as follows. In Section 2, we give some background on multiagent systems policies along with two multiagent system examples. In Section 3, we define the notion of *system traces* for a multiagent system, which are later used to describe policies. Section 4 defines *law policies* as well as *guidance policies*; we give examples and show how *guidance policies* are useful for multiagent systems and describe a method for ordering guidance policies according to importance. Section 5 presents and analyzes experimental results from applying policies to the two multiagent system examples. Section 6 concludes and presents some future work.

2 Background

Policies have been considered for multiagent systems for some time. Efforts have been made to characterize, represent, and reason [2] about policies in the context of multiagent systems. Policies have been referred to as laws in the past. Yoav Shoham and Moshe Tennenholtz wrote in [3] about *social laws* for multiagent systems. They showed how policies could help a system to work together, similar to how our rules of driving on a predetermined side of the road help the traffic to move smoothly.There has also been work on detecting global properties [4] of a distributed system, which could in turn be used to suggest policies for that system. Policies have also been proposed as a way to help assure that agents and that the entire multiagent system behave within certain boundaries. They have also been proposed as a way to specify security constraints in multiagent systems [5, 6]. There has been work to define policy languages by defining a description logic [7]. Policies have also been referred to as *norms*. Much work has been done on the formal specification of these norms [8]. We are taking this formal approach in our specification of guidance and law policies. Norms, however, are usually associated with *open systems*–while we are concerned with *closed, cooperative systems*. We want to use formal methods to prove whether a given system will abide by the policies as expected. Thus, we must give our guidance policies for multiagent societies a solid formal foundation. In order to achieve this end, we borrow concepts that are widely used in program analysis, in particular, model checking. Taking a model checking approach to policies has been done [9] and is a natural extension of program analysis.

The multiagent systems model we are using for this paper is called the Organization Model for Adaptive Computational Systems (OMACS) [10]. Figure 1 is a graphical depiction of the OMACS model. OMACS defines standard multiagent system components such as goals, roles, capabilities, and agents. Roles *achieve* goals, agents *posses* capabilities, and agents are *capable* of playing roles depending on the capabilities they

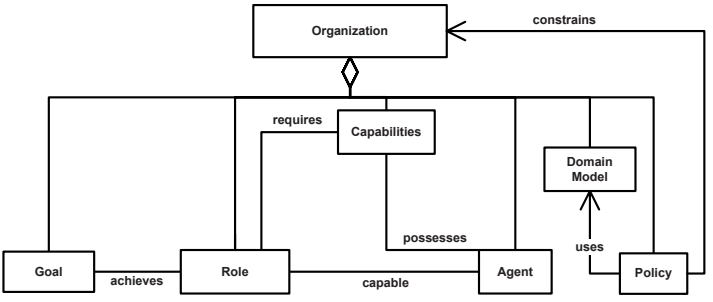


Fig. 1. Organization Model for Adaptive Computational Systems

posses. The organization, which represents the entire set of agents, decides which agents to *assign* to what roles to *achieve* particular goals. When the organization makes an *assignment* of an agent to a particular role to achieve a specific goal, the organization is constrained by agents capabilities as well as any applicable policies. To model goals, we use the Goal Model for Dynamic Systems (GMoDS) as defined in [11]. Events may occur while an agent is playing a role. These events may *trigger* (activate) goals. Only active goals may be assigned to an agent.

2.1 Conference Management Example

A well known example in multiagent systems is the Conference Management [12,13] example. The Conference Management example models the workings of a scientific conference, for example, authors submit papers, reviewers review the submitted papers, and certain papers are selected for the conference and printed in the proceedings. Figure 2 shows the complete goal model for the conference management example, which we are using to illustrate our policies. In this example, a multiagent system represents the goals and tasks of a generic conference paper management system. Goals of the system are identified and are decomposed into subgoals.

The top-level goal, *0. Manage conference submissions*, is decomposed into several "*and*" subgoals, which means that in order to achieve the top goal, the system must achieve all of its subgoals. These subgoals are then associated through precedence and trigger relations. The *precedes* arrow between goals indicates that the source of the arrow must be *achieved* before the destination can become active. The *triggers* arrow indicates that the domain-specific event in the source may trigger the goal in the destination. The *occurs* arrow from a goal to a domain-specific event indicates that while pursuing that goal, said event may occur. A goal that triggers another goal may trigger multiple instances of that goal.

Leaf goals are goals that have no children. The leaf goals in this example consist of *Collect papers*, *Distribute papers*, *Partition papers*, *Assign reviewers*, *Collect reviews*, *Make decision*, *Inform accepted*, *Inform declined*, *Collect finals*, and *Send to printer*. For each of these leaf goals to be achieved, agents must play specific roles. The roles required to achieve the leaf goals are depicted in Figure 3. The role model gives seven roles as well as two outside actors. Each role contains a list of leaf goals that the role can achieve. For example, the *Assigner* role can achieve the *Assign reviewers* leaf goal.

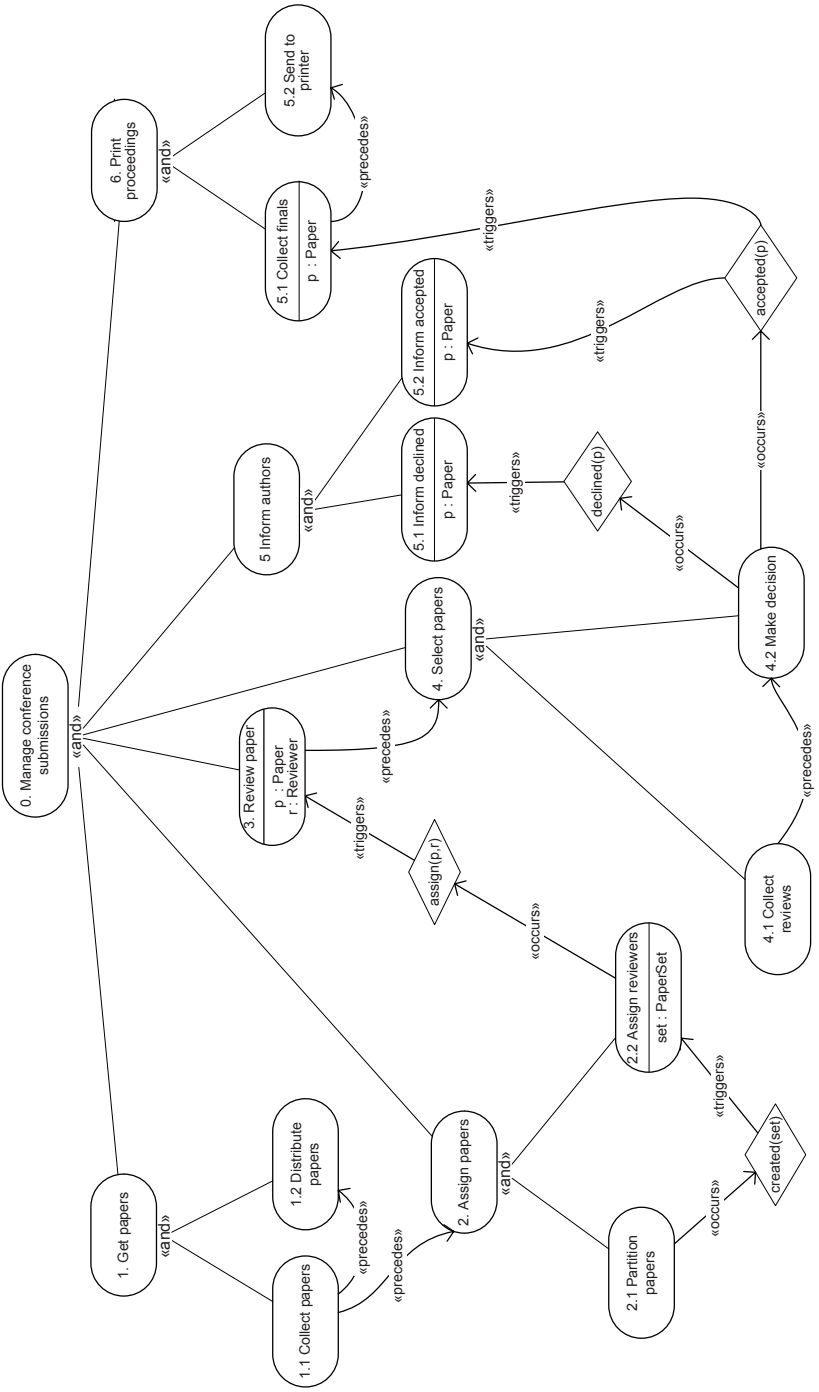


Fig. 2. Conference Management Goal Model

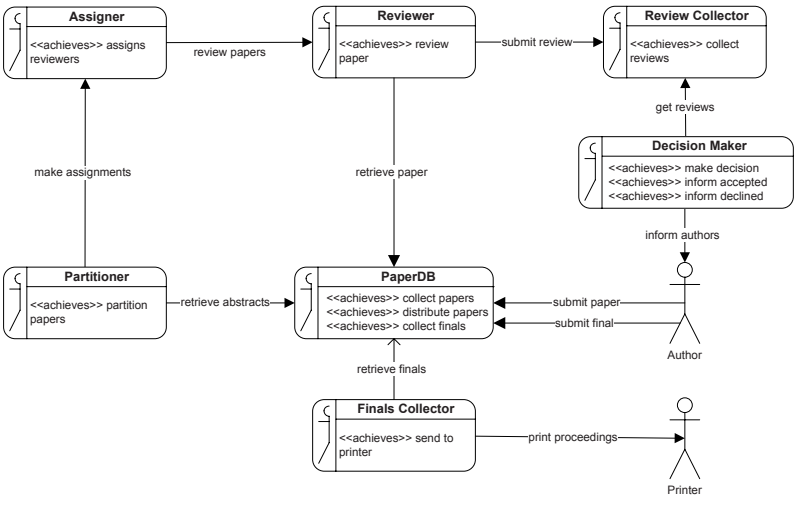


Fig. 3. Conference Management Role Model

In GMoDS, roles only achieve leaf goals. The arrows between the roles indicates interaction between particular roles. For example, once the agent playing the *Partitioner* role has some partitions, it will need to hand off these partitions to the agent playing the *Assigner* role. OMACS allows an agent to play multiple roles simultaneously, as long as it has the capabilities required by the roles and it is allowed by the policies.

2.2 Robotic Floor Cleaning Example

Another example to illustrate the usefulness of the concept of guidance policies is the Cooperative Robotic Floor Cleaning Company Example (CRFCC), which was first presented by Robby et al. in [14]. In this example, a team of robotic agents clean the floors of a building. The team has a map of the building as well as indications of whether a floor is tile or carpet. Each team member will have a certain set of capabilities (e.g. vacuum, mop, etc). These capabilities may become defective over time. In their analysis, Robby et al. showed how breaking up the capabilities affected a team's flexibility to overcome loss of capabilities. We have extended this example by giving the information that the vacuum cleaner's bag needs to be changed after vacuuming three rooms. Thus, we want to minimize the number of bag changes. For this, we introduce a guidance policy and show how it affects the performance of the organization.

The goal model for the CRFCC system is fairly simple. As seen in Figure 4, the overall goal of the system (Goal 0) is to clean the floors. This goal is decomposed into three conjunctive subgoals: *1. Divide Area*, *2. Pickup*, and *3. Clean*. The *3. Clean* goal is decomposed into two disjunctive goals: *3.1 Sweep & Mop* and *3.2 Vacuum*. Depending on the floor type, only one subgoal must be achieved to accomplish the *3. Clean* goal. If an area needs to be swept and mopped (i.e. it is tile), then goal *3.1 Sweep & Mop* is decomposed into two conjunctive goals: *3.1.1 Sweep* and *3.1.2 Mop*. After an agent achieves the *1. Divide area* goal, a certain number of *2. Pickup* goals will become active

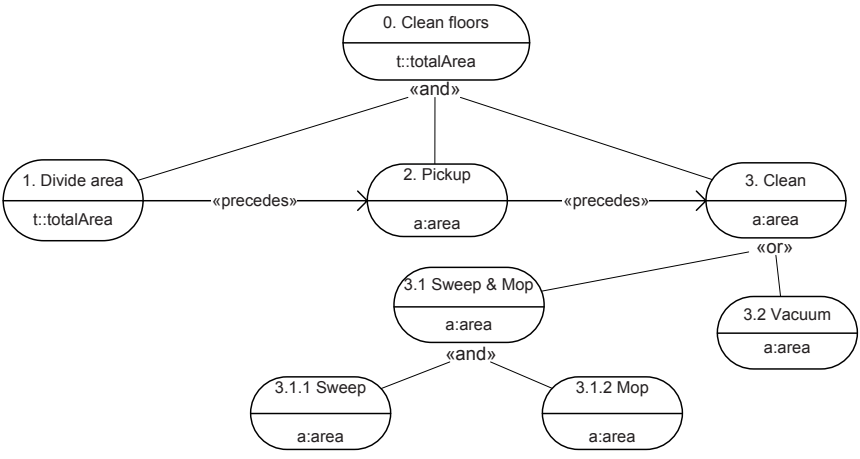


Fig. 4. CRFCC Goal Model

Role Name	Req. Capabilities	Goals Achieved
Organizer	org	1. Divide Area
Pickuper	search, move	2. Pickup
Sweeper	sweep	3.1.1 Sweep
Mopper	mop	3.1.2 Mop
Vacuummer	vacuum	3.2 Vacuum

Fig. 5. CRFCC Role Model

(depending on how many pieces the area is divided into). After the *2. Pickup* goals are completed, a certain number of *3. Clean* goals become active, again depending on how many pieces the area was broken into. This then will activate goals for the tile areas (*3.1.1 Sweep* and *3.1.2 Mop*) as well as goals for the carpeted areas (*3.2 Vacuum*).

Figure 5 gives the role model for the CRFCC. In this role model, each leaf goal of the system is achieved by a specific role. The role model may be designed many different ways depending on the system's goal, agent, and capability models. Thus, depending on the agents and capabilities available, the system designer may choose different role models. For this paper, we will look at just one of these possible role models. In the role model in Figure 5, the only role requiring more than one capability is the *Pickuper* role. This role will require both the *search* and *move* capability. Thus, in order to play this role, an agent must possess both capabilities.

3 Multiagent Traces

There are several observable events in an OMACS system. A *system event* is simply an action taken by the system. In this paper, we are concerned with specific actions that the organization takes. For instance, an assignment of an agent to a role is a system event.

Event	Definition
$C(g_i)$	goal g_i has been completed.
$T(g_i)$	goal g_i has been triggered.
$A(a_i, r_j, g_k)$	agent a_i has been assigned role r_j to achieve goal g_k.

(a) System Events

Property	Definition
$a.reviews$	the number of reviews agent a has performed.
$a.vacuumedRooms$	the number of rooms agent a has vacuumed.

(b) Properties

Fig. 6. Events and Properties of Interest

The completion of a goal is also a system event. In an OMACS system, we can have the system events of interest shown in Figure 6(a).

At any stage in a multiagent system, there may be certain properties of interest. Some may be domain-specific (only relevant to the current system), while others may be general properties such as the number of roles an agent is currently playing. State properties that are relevant to the examples we are presenting in the next section are shown in Figure 6(b).

3.1 System Traces

In order to describe multiagent system execution, we use the notion of a system trace. An (abstract) *system trace* is a projection of system execution with only desired state and event information preserved (role assignments, goal completions, domain-specific state property changes, etc). In this paper, we are only concerned with the events and properties given above and only traces that result in a successful completion of the system goal. Let E be an event of interest and P be a property of interest. A *change of interest* in a property is a change for which a system designer has made some policy. For example, if a certain integer should never exceed 5, a change of interest would be when that integer became greater than 5 and when that integer became less than 5. Thus a change of interest in a property is simply an abstraction of all the changes in the property. ΔP indicates a change of interest in property P. A system trace may contain both events and changes of interest in properties. Changes of interest in properties may be viewed as events, however, for simplicity we include both and use both interchangeably. Thus, a system trace is defined as:

$$E_1 \rightarrow E_2 \rightarrow \dots \tag{1}$$

As shown in equation 1, a trace is simply a sequence of events. An example subtrace of a multiagent system, where g_1 is a goal, a_1 is an agent, and r_1 is a role, might be:

$$\dots T(g_1) \rightarrow A(a_1, r_1, g_1) \rightarrow C(g_1) \dots \tag{2}$$

Formula 2 means that goal g_1 is triggered, then agent a_1 is assigned role r_1 to achieve goal g_1, finally, goal g_1 is completed.

We use the terms *legal trace* and *illegal trace*. An *illegal trace* is an execution we do not want our system to exhibit, while a *legal trace* is an execution that our system may exhibit. Intuitively, policies cause some traces to become *illegal*, while others remain *legal*.

We are able to use the notion of system traces because the framework we are using to build multiagent systems constructs mathematically specified models (e.g [10,11]) of various aspects of the system (goal model, role model, etc.). This can be leveraged to formally specify policies as restrictions of system traces. Once we have a formal definition of system traces, we can leverage existing research on property specification and concurrent program analysis.

4 Policies

Policies may restrict or proscribe behaviors of a system. Policies concerning agent assignments to roles have the effect of constraining the set of possible assignments. This can greatly reduce the search space when looking for the optimal assignment set [15].

Other policies can be used for verifying that a goal model meets certain criteria. This allows the system designer to more easily state properties of the goal model that may be verified against candidate goal models at design time. For example, one might want to ensure that our goal model in Figure 2 will always trigger a *Review Paper* goal for each paper submitted.

Yet, other policies may restrict the way that roles can be played. For example, *when an agent is moving down the sidewalk it always keeps to the right.* These behavior policies also restrict how an agent interacts with its environment, which in turn means that they can restrict protocols and agent interactions. One such policy might be that an agent playing the *Reviewer* role must always give each review a unique number. These sort of policies rely heavily on domain-specific information. Thus it is important to have an ontology for relevant state and event information prior to designing policies [16].

4.1 Language for Policy Analysis

To describe our policies, we use temporal formula with quantification similar to [17]. This may be converted into Linear Temporal Logic (LTL) [18] or Büchi automata [19] for infinite system traces, or to something like Quantified Regular Expressions [20] for finite system traces. The formulas consist of predicates over goals, roles, events, and assignments (recall that an assignment is the joining of an agent and role for the purpose of achieving a goal). The temporal operators we currently use are as follows: $\Box(x)$, meaning x holds always; $\Diamond(x)$, meaning x holds eventually; and $x\,\mathcal{U}\,y$, meaning x holds until y holds.[1] We use a mixture of state properties as well as events [21] to obtain compact and readable policies. An example of one such policy formula is:

$$\forall a_1 : Agents,\ \mathcal{L} : \Box(sizeOf(a_1.reviews) \leq 5) \tag{3}$$

Formula 3 states that it should always be the case that each agent never review more than five papers. The $\mathcal{L}$: indicates that this is a *law policy*. The property $.reviews$ can be considered as part of the system's state information. This is domain-specific and allows a more compact representation of the property. This policy may be easily represented by a finite automata as shown in Figure 7.

[1] We only reason about bounded liveness properties because we only consider successful traces.

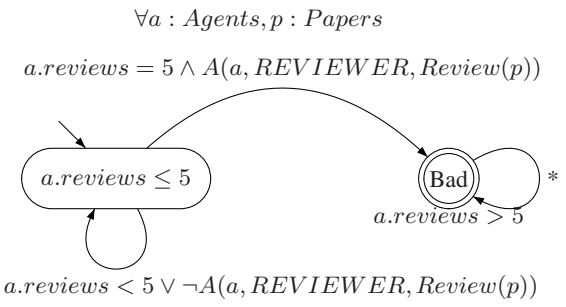


Fig. 7. No agent may review more than five papers

The use of the $A()$ predicate in Figure 7 indicates an assignment of the *Reviewer* role to achieve the *Review paper* goal, which is parametrized on the paper p. This automata depicts the policy in Formula 3, but in a manner for a model checker or some other policy enforcement mechanism to detect when violation occurs. The accepting state indicates that a violation has occurred. Normally, this automata would be run alongside the system, either at design time with a model checker [22], or at run-time with some policy enforcement mechanism [23].

We would like to emphasize here that we do not expect the designer to specify their policies by hand editing LTL. LTL is complex and designing policies in LTL would be very error prone and thus could potentially mislead the designer into a false sense of security or simply compose incorrect policies. There has been some work in facilitating the creation of properties in LTL (and other formalisms) for program analysis such as specification patterns [24]. There has also been work done to help system property specification writers to graphically create properties [25] (backed by LTL) by manipulating automata and answering simple questions regarding elements of the property.

4.2 Law Policies

The traditional notion of a policy is a rule that must always be followed. We refer to these policies as *law policies*. An example of a law policy with respect to our conference management example would be *no agent may review more than five papers.* This means that our system can never assign an agent to the *Reviewer* role more than five times. A law policy can be defined as:

$$\mathcal{L} : Conditions \rightarrow Property \tag{4}$$

$Conditions$ are predicates over state properties and events, which, when held true, imply that the $Property$ holds true. The $Conditions$ portion of the policy may be omitted if the $Property$ portion should hold in all conditions, as in Formula 3.

Intuitively, for the example above, no trace in the system may contain a subtrace in which an agent is assigned to the *Reviewer* role more than five times. This will limit the number of legal traces in the system. In general, *law policies reduce the number of legal traces for a multiagent system.* The policy to limit the number of reviews an agent can perform is helpful in that it will ensure that our system does not overburden any agent

with too many papers to review. This policy as a pure law policy, however, could lead to trouble in that the system may no longer be able to achieve its goal. Imagine that more papers than expected are submitted. If there are not sufficient agents to spread the load, the system will fail since it is cannot assign more than five papers to any agent. This is a common problem with using only law policies. They limit the *flexibility* of the system, which we define as *how well the system can adapt to changes* [14].

4.3 Guidance Policies

While the policy in (3) is a seemingly useful policy, it reduces flexibility. To overcome this problem, we have defined another, weaker type of policy called *guidance policies*. Take for example the policy used above, but as a *guidance policy*:

$$\forall a_1 : Agents,\ \mathcal{G} :\Box(sizeOf(a_1.reviews) \leq 5) \tag{5}$$

This is the same as the policy as in (3) except for the $\mathcal{G}$:, which indicates that it is a *guidance policy*. In essence, the formalization for guidance and law policies are the same, the difference is the intention of the system designer. *Law policies* should be used when the designer wants to make sure that some property is always true (e.g. for safety or security), while *guidance policies* should be used when the designer simply wants to guide the system. This guidance policy limits our agents to reviewing no more than five papers, *when possible*. Now, the system can still be successful when it gets more submissions than expected since it can assign more than five papers to an agent. When there are sufficient agents, the policy still limits each agent to five or fewer reviews.

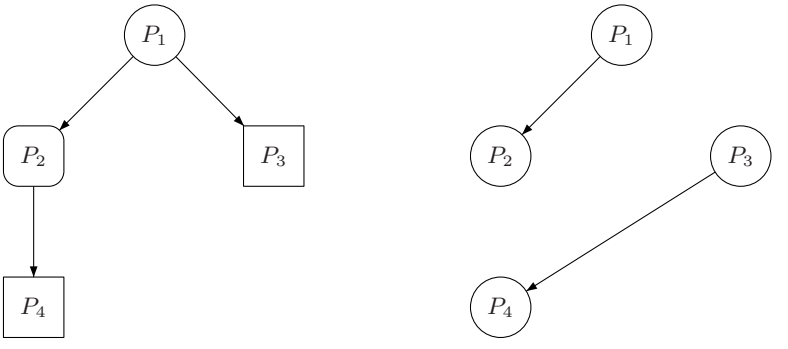
Guidance policies more closely emulate how policies are implemented in human societies. They also provide a clearer and simpler construct for more easily and accurately describing the design of a multiagent organization. In contrast to policy resolution complexity of detecting and resolving policy contradictions in [2], our methodology of using guidance policies present an incremental approach to policy resolution. That is, the system will still work under conflicting policies; its behaviors are amenable to analysis, thus allowing iterative policy refinement.

In the definition of *guidance policies*, we have not specified how the system should choose which guidance policy to violate in a given situation. We propose a partial ordering of guidance policies to allow the system designer to set precedence relationships between guidance policies. We arrange the guidance policies as a lattice, such that a policy that is a parent of another policy in the lattice, is *more-important-than* its children. By analyzing a system trace, one can determine a set of policies that were violated during that trace. This set of violations may be computed by examining the policies and checking for matches against the trace. When there are two traces that violate policies with a common ancestor, and one (and only one) of the traces violate the common ancestor policy, we mark the trace violating that common ancestor policy as illegal. Intuitively, this trace is illegal because the system could have violated a less important policy. Thus, if the highest policy node violated in each of the two traces is an ancestor of every node violated in both traces, and that node is not violated in both traces, then we know the trace violating that node is illegal and should not have happened.

Take, for example, the four policies in the Table 1. Let these policies be arranged in the lattice shown in Figure 8(a). The lattice in Figure 8(a) means that policy P_1 is more

Table 1. Conference Management Policies

Node	Definition
P_1	No agent should review more than 5 papers.
P_2	PC Chair should not review papers.
P_3	Each paper should receive at least 3 reviews.
P_4	An agent should not review a paper from someone whom they wrote a paper with.



(a) Possible Partial order of Guidance Policies.

(b) Another possible ordering.

Fig. 8. Partial orders of Guidance Policies

important than P_2 and P_3, and P_2 is more important than P_4. Thus, if there is any trace that violates any guidance policies other than P_1 (and does not violate a law policy), it should be chosen over one which violates P_1.

When a system cannot achieve its goals without violating policies, it may violate guidance policies. There may be traces that are still illegal, though, depending on the ordering between policies. *For every pair of traces, if the least upper bound of the policies violated in both traces, let us call this policy violation $\mathcal{P}$, is in one (and only one) of the traces, the trace with $\mathcal{P}$ is illegal.* For example, consider the ordering in Figure 8(a), let trace t_1 violate P_1 and P_2, while trace t_2 violates P_2 and P_3. Round nodes represent policies violated in t_1, box nodes represent policies violated in t_2, and boxes with rounded corners represent policies violated in both t_1 and t_2. Since P_1 is the least upper bound of P_1, P_2, and P_3 and since P_1 is not in t_2, t_1 is illegal.

As shown in Figure 8(b), the policies may be ordered in such a way that the policy violations of two traces do not have a least upper bound. If there is no least upper bound, $\mathcal{P}$, such that $\mathcal{P}$ is in one of the traces, the two traces cannot be compared and thus both traces are legal. The reason they cannot be compared is that we have no information about which policies are more important. Thus, either option is legal. It is important to see here that all the guidance policies do not need to be ordered into a single lattice. The system designer could create several unrelated lattices. These lattices then can be iteratively refined by observing the system behaviors or by looking at metrics generated

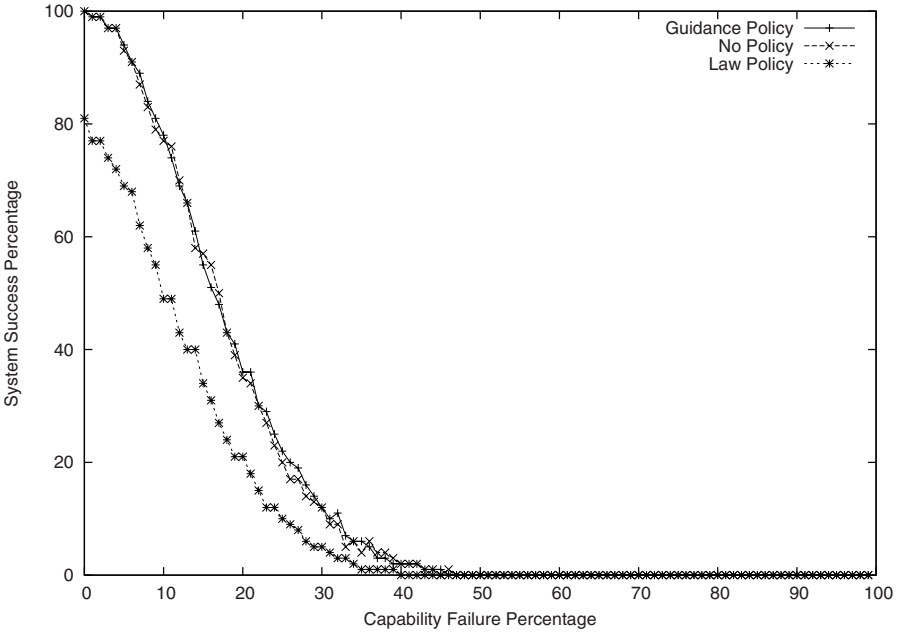


Fig. 9. The success rate of the system given capability failure

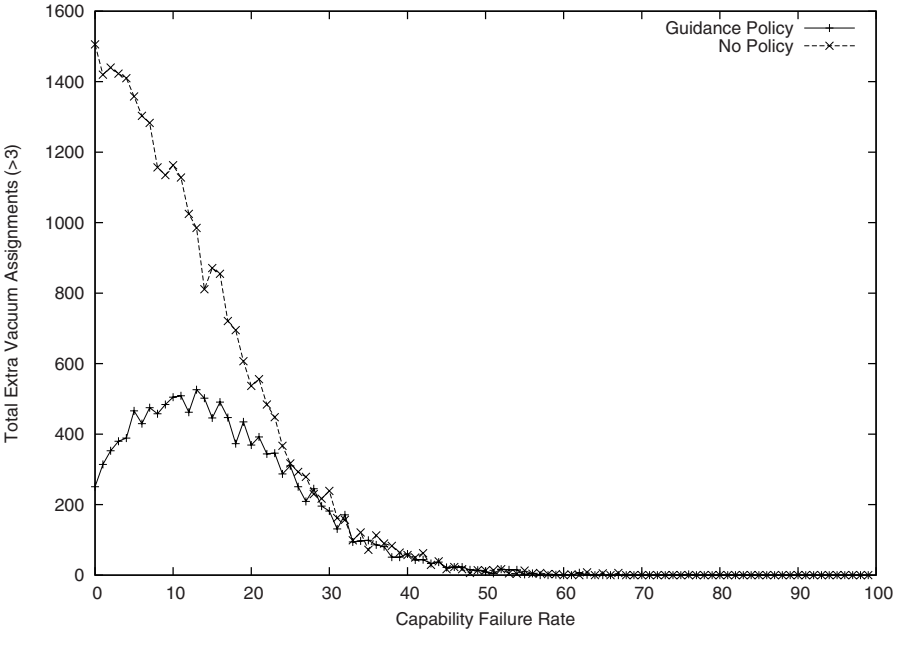


Fig. 10. The extra vacuum assignments given capability failure

for a certain policy set and ordering (e.g., [14]). This allows the system designer to influence the behavior of the system by making logical choices as to what paths are considered better. Using the lattice in Figure 8(a), we may even have the situation where P_1 is not violated by either trace. In this case, the violation sets cannot be compared, and thus, both traces are legal. In situations such as these, the system designer may want to impose more ordering on the policies.

Intuitively, guidance policies constrain the system such that at any given state, transitions that will not violate a guidance policy are always chosen over transitions that violate a guidance policy. If guidance policy violation cannot be avoided, a partial ordering of guidance policies is used to choose which policies to violate.

5 Evaluation

5.1 CRFCC

Using our CRFCC example and a modified simulator from [14], we collected results running simulations with the guidance policy: *no agent should vacuum more than three rooms*. We contrast this with the law policy: *no agent may vacuum more than three rooms*. The guidance policy is presented formally in Equation 6.

$$\forall a_1 : Agents,\ \mathcal{G} :\Box(a_1.vacuumedRooms \leq 3) \tag{6}$$

For this experiment, we used five agents each having the following capabilities: a_1, org, search, and move; a_2, search, move, and vacuum; a_3, vacuum and sweep; a_4, sweep and mop; and a_5, org and mop. These capabilities restrict the roles our simulator can assign to particular agents. For example, the Organizer role may only be played by agent a_1 or agent a_5, since those are the only agents with the *org* capability. In the simulation we randomly choose capabilities to fail based on a probability given by the *capability failure rate*.

For each experiment, the result of 1000 runs at each capability failure rate was averaged. At each simulation step, a goal being played by an agent is randomly achieved. Using the capability failure rate, at each step, a random capability from a random agent may be selected to fail. Once a capability fails it cannot be repaired.

Figure 9 shows that while the system success rate decreases when we enforce the law policy, it does not, however, decrease when we enforce the guidance policy. Figure 10 shows the total number of times the system assigned vacuuming to an agent who already vacuumed at least 3 rooms for 1000 runs of the simulation at each failure rate. With no policy, it can be seen that the system will in fact assign an agent to vacuum more than 3 rooms quite often. With the guidance policy, however, the extra vacuum assignments (> 3) stay minimal. The violations of the guidance policy increase as the system must adapt to an increasing failure of capabilities until it reaches a peak. At the peak, increased violations do not aid in goal achievement and eventually the system cannot succeed even without the policy. Thus, the system designer may now wish to purchase equipment with a lower rate of failure, or add more redundancy to the system to compensate. The system designer may also evaluate the graph and determine whether the cost of the maximum number of violations exceeds the maximum cost he is willing to incur, and if not, make appropriate adjustments.

5.2 Conference Management System

We also simulated the conference management system described in Section 2.1. We held the number of agents constant, while increasing the number of papers submitted to the conference. The system was constructed with a total of 13 agents, 1 *PC Member* agent, 1 *Database* agent, 1 *PC Chair* agent, and 10 *Reviewer* agents. The simulation randomly makes goals available to achieve, while still following the constraints imposed by GMoDS. Roles that achieve the goal are chosen at random as well as agents that can play the given role. The policies are given priority using the *more-important-than* relation as depicted in Figure 8(a).

Figure 11 shows a plot of how many times a guidance policy is violated versus the number of papers submitted for review. For each set of paper submissions (from 1 to 100) we ran the simulation 1000 times and then took the average of the 1000 runs to determine the average number of violations. In all the runs the system succeeded in achieving the top level goal.

As seen by the graph in Figure 11, no policies are violated until around 17 papers (this number is explained below). The two least important policies (P_2 and P_3) are violated right away. The violation of P_2, however, levels off since it is interacting with P_1. The violations of P_3 is seen to grow at a much greater rate since it is the least important policy.

We then changed all the guidance policies to law policies and re-ran the simulation. For 17 or more submissions, the system always failed to achieve the top level goal. This makes sense because we have only 10 Reviewer agents and we have the policies: the

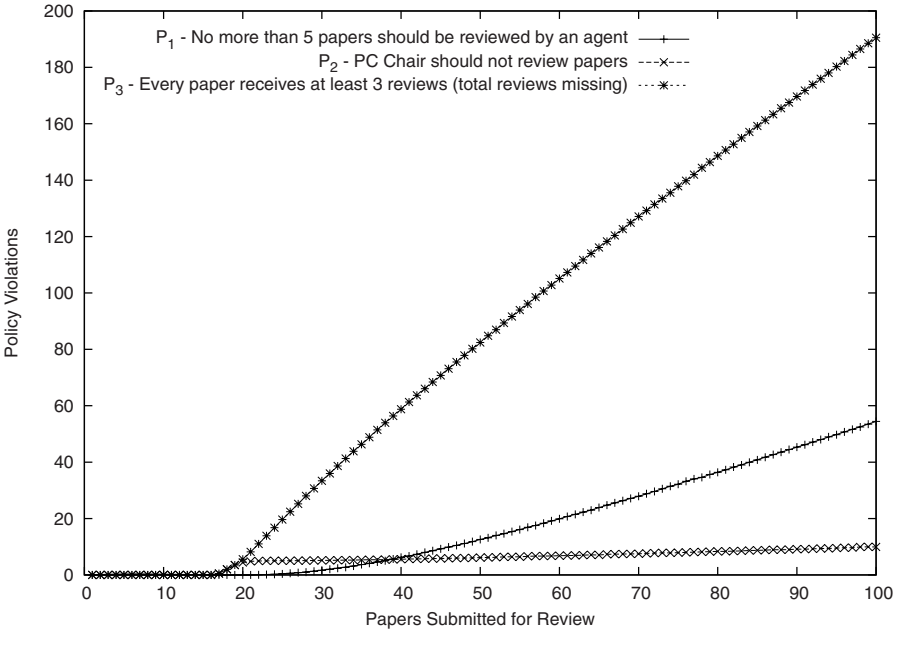


Fig. 11. Violations of the guidance policies as the number of papers to review increases

PC Chair should not review papers and no agent should review more than 5 papers. This means the system can only produce $5 \times 10 = 50$ reviews. But, since we have the policy that each paper should have at least 3 reviews, 17 submissions would need $17 \times 3 = 51$ reviews. For 16 or fewer papers submitted, the law policies perform identical to the guidance policies.

5.3 Common Results

As the experimental results in Figure 9 show, guidance policies *do not decrease the flexibility of a system to adapt to a changing environment*, while law policies *do decrease the flexibility of a system to adapt to a changing environment*. Guidance policies, however, do help guide the system and improve performance as shown in Figure 10 and Figure 11. The partial ordering using the *more-important-than* relation helps a system designer put priorities on what policies they consider to be more important and helps the system decide which policies to violate in a manner consistent with the designer's intentions.

6 Conclusions and Future Work

Policies have proven to be useful in the development of multiagent systems. However, if implemented inflexibly, situations such as described in [26] will occur (a policy caused a spacecraft to crash into an asteroid). Guidance policies allow a system designer to guide the system while giving it a chance to adapt to new situations.

With the introduction of guidance policies, policies are an even better mechanism for describing desired properties and behaviors of a system. It is our belief that guidance policies more closely capture how policies work in human organizations. Guidance policies allow for more flexibility than law policies in that they may be violated under certain circumstances. In this paper, we demonstrated a technique to resolve conflicts when faced with the choice of which guidance policies to violate. Guidance policies, since they may be violated, can have a partial ordering. That is, one policy may be considered more important than another. In this manner, we allow the system to make better choices on which policies to violate. Traditional policies may be viewed as *law policies*, since they must never be violated. Law policies are still useful when the system designer never wants a policy to be violated–regardless of system success. Such policies might concern security or human safety.

Policies may be applied in an OMACS system by constraining assignments of agents to roles, the structure of the goal model for the organization, or how the agent may play a particular role. Through the use of OMACS, the metrics described in [14], and the policy formalisms presented here, we are able to provide an environment in which a system designer may formally evaluate a candidate design, as well as evaluate the impact of changes to that design without deploying or even completely developing the system.

Policies can dramatically improve run-time of reorganization algorithms in OMACS as shown in [15]. Guidance policies can be a way to achieve this run-time improvement without sacrificing system flexibility. The greater the flexibility, the better the chance that the system will be able to achieve its goals.

Policies are an important part of a multiagent system. Future work is planned to ease the expression and analysis of policies. Some work has already been done in this area

[24, 25], but it has not been integrated with a multiagent system engineering framework. Another area of work is to provide a verification framework from design all the way to implementation. The goal would be to determine the minimum guarantees needed from the agents to guarantee the overall system behavior specified by the policies. These minimum guarantees could then be checked against the agent implementations to determine whether the implemented system follows the policies given.

Guidance policies add an important tool to multiagent policy specification. However, with this tool comes complexity. Care must be taken to insure that the partial ordering given causes the system to exhibit the behavior intended. Tools which can visually depict the impact of orderings would be helpful to the engineer considering various orderings. We are currently working on inferring new policies from a given set of policies. For example, if a system designer wanted to get their system to a state for which they defined policy, we would automatically generate guidance policies. This could be useful when the policies are defined as finishing moves in chess. That is they proscribe optimal behavior, given a state. Thus, we would like to get to the state where we know that optimal behavior. Another exciting area of research is to determine a method of dynamically learning guidance policies, which would allow an organization to evolve within its changing environment.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
2. Bradshaw, J., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M., Acquisti, A., Benyo, B., Breedy, M., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., Van Hoof, R.: Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 835–842. ACM Press, New York (2003)
3. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: Off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
4. Stoller, S.D., Unnikrishnan, L., Liu, Y.A.: Efficient detection of global properties in distributed systems using partial-order methods. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 264–279. Springer, Heidelberg (2000)
5. Kagal, L., Finin, T., Joshi, A.: A policy based approach to security for the semantic web. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 402–418. Springer, Heidelberg (2003)
6. Paruchuri, P., Tambe, M., Ordóñez, F., Kraus, S.: Security in multiagent systems by policy randomization. In: AAMAS 2006: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 273–280. ACM Press, New York (2006)
7. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: Kaos policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement. In: POLICY 2003: IEEE 4th International Workshop on Policies for Distributed Systems and Networks, pp. 93–96. IEEE, Los Alamitos (2003)
8. Artikis, A., Sergot, M., Pitt, J.: Specifying norm-governed computational societies. ACM Transactions on Computational Logic (2007)

9. Viganò, F., Colombetti, M.: Symbolic Model Checking of Institutions. In: Proceedings of the 9th International Conference on Electronic Commerce (2007)
10. DeLoach, S.A., Oyenan, W., Matson, E.T.: A capabilities based theory of artificial organizations. Journal of Autonomous Agents and Multiagent Systems (2007)
11. Miller, M.: A goal model for dynamic systems. Master's thesis, Kansas State University (April 2007)
12. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Organisational rules as an abstraction for the analysis and design of multi-agent systems. International Journal of Software Engineering and Knowledge Engineering 11(3), 303–328 (2001)
13. DeLoach, S.A.: Modeling organizational rules in the multi-agent systems engineering methodology. In: Cohen, R., Spencer, B. (eds.) Canadian AI 2002. LNCS (LNAI), vol. 2338, pp. 1–15. Springer, Heidelberg (2002)
14. Robby, DeLoach, S.A., Kolesnikov, V.A.: Using design metrics for predicting system flexibility. In: Baresi, L., Heckel, R. (eds.) FASE 2006 and ETAPS 2006. LNCS, vol. 3922, pp. 184–198. Springer, Heidelberg (2006)
15. Zhong, C., DeLoach, S.A.: An investigation of reorganization algorithms. In: Proceedings of the International Conference on Artificial Intelligence (ICAI 2006), Las Vegas, Nevada, pp. 514–517. CSREA Press (June 2006)
16. DiLeo, J., Jacobs, T., DeLoach, S.: Integrating ontologies into multiagent systems engineering. In: Fourth International Conference on Agent-Oriented Information Systems (AIOS 2002), CEUR-WS.org (July 2002)
17. Corbett, J.C., Dwyer, M.B., Hatcliff, J., Robby: Expressing checkable properties of dynamic systems: The bandera specification language. International Journal on Software Tools for Technology Transfer (STTT) 4(1), 34–56 (2002)
18. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer, Heidelberg (1991)
19. Büchi, J.R.: On a decision method in restricted second-order arithmetics. In: Proceedings of International Congress of Logic Methodology and Philosophy of Science, Palo Alto, CA, USA, pp. 1–12. Stanford University Press (1960)
20. Olender, K.M., Osterweil, L.J.: Cecil: A sequencing constraint language for automatic static analysis generation. IEEE Transactions on Software Engineering 16(3), 268–280 (1990)
21. Chaki, S., Clarke, E.M., Ouaknine, J., Sharygina, N., Sinha, N.: State/event-based software model checking. In: Boiten, E.A., Derrick, J., Smith, G.P. (eds.) IFM 2004. LNCS, vol. 2999, pp. 128–147. Springer, Heidelberg (2004)
22. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
23. Ligatti, J., Bauer, L., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policies. In: International Journal of Information Security, vol. 4, pp. 2–16. Springer, Heidelberg (2004)
24. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 1999 International Conference on Software Engineering. IEEE, Los Alamitos (1999)
25. Smith, R.L., Avrunin, G.S., Clarke, L.A., Osterweil, L.J.: Propel: an approach supporting property elucidation. In: ICSE 2002: Proceedings of the 24th International Conference on Software Engineering, pp. 11–21. ACM Press, New York (2002)
26. Peña, J., Hinchey, M.G., Sterritt, R.: Towards modeling, specifying and deploying policies in autonomous and autonomic systems using an AOSE methodology. EASE 0, 37–46 (2006)

Author Index