Logic and the Simulation of Interaction and Reasoning

AISB '08

AISB 2008 Proceedings Volume 9

GLoRiClass

UNIVERSITY OF ABERDEEN

AISB 2008 Convention
Communication, Interaction and Social Intelligence

1st-4th April 2008

University of Aberdeen

Volume 9:
Proceedings of the
AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning

Published by
The Society for the Study of Artificial Intelligence and Simulation of Behaviour

http://www.aisb.org.uk/convention/aisb08/

ISBN 1 902956 68 0

Contents

The AISB'08 Convention: Communication, Interaction and Social Intelligence

As the field of Artificial Intelligence matures, AI systems begin to take their place in human society as our helpers. Thus it becomes essential for AI systems to have sophisticated social abilities, to communicate and interact. Some systems support us in our activities, while others take on tasks on our behalf. For those systems directly supporting human activities, advances in human-computer interaction become crucial. The bottleneck in such systems is often not the ability to find and process information; the bottleneck is often the inability to have natural (human) communication between computer and user. Clearly such AI research can benefit greatly from interaction with other disciplines such as linguistics and psychology. For those systems to which we delegate tasks: they become our electronic counterparts, or agents, and they need to communicate with the delegates of other humans (or organisations) to complete their tasks. Thus research on the social abilities of agents becomes central, and to this end multi-agent systems have had to borrow concepts from human societies. This interdisciplinary work borrows results from areas such as sociology and legal systems. An exciting recent development is the use of AI techniques to support and shed new light on interactions in human social networks, thus supporting effective collaboration in human societies. The research then has come full circle: techniques which were inspired by human abilities, with the original aim of enhancing AI, are now being applied to enhance those human abilities themselves. All of this underscores the importance of communication, interaction and social intelligence in current Artificial Intelligence and Cognitive Science research.

In addition to providing a home for state-of-the-art research in specialist areas, the convention also aimed to provide a fertile ground for new collaborations to be forged between complementary areas. Furthermore the 2008 Convention encouraged contributions that were not directly related to the theme, notable examples being the symposia on "Swarm Intelligence" and "Computing and Philosophy".

The invited speakers were chosen to fit with the major themes being represented in the symposia, and also to give a cross-disciplinary flavour to the event; thus speakers with Cognitive Science interests were chosen, rather than those with purely Computer Science interests. Prof. Jon Oberlander represented the themes of affective language, and multimodal communication; Prof. Rosaria Conte represented the themes of social interaction in agent systems, including behaviour regulation and emergence; Prof. Justine Cassell represented the themes of multimodal communication and embodied agents; Prof. Luciano Floridi represented the philosophical themes, in particular the impact on society. In addition there were many renowned international speakers invited to the individual symposia and workshops. Finally the public lecture was chosen to fit the broad theme of the convention – addressing the challenges of developing AI systems that could take their place in human society (Prof. Aaron Sloman) and the possible implications for humanity (Prof. Luciano Floridi).

The organisers would like to thank the University of Aberdeen for supporting the event. Special thanks are also due to the volunteers from Aberdeen University who did substantial additional local organising: Graeme Ritchie, Judith Masthoff, Joey Lam, and the student volunteers. Our sincerest thanks also go out to the symposium chairs and committees, without whose hard work and careful cooperation there could have been no Convention. Finally, and by no means least, we would like to thank the authors of the contributed papers – we sincerely hope they get value from the event.

Frank Guerin & Wamberto Vasconcelos

Logic and the Simulation of Interaction and Reasoning: Introductory Remarks.

Benedikt Löwe [1]

Abstract. This introductory note provides the background for the symposium *"Logic and the Simulation of Interaction and Reasoning"*, its motivations and the 15 papers presented at the symposium.

1 INTRODUCTION

In the past years, logicians have become more and more interested in the phenomenon of interaction. The area "*Logic and Games*" deals with the transition from the static logical paradigm of formal proof and derivation to the dynamic world of intelligent interaction and its logical models. A number of conferences and workshops such as the LOFT ("Logic and the Foundations of Game and Decision Theory") series, the 7th Augustus de Morgan Workshop in London [November 2005; [1]], the Royal Academy Colloquium 'New perspectives on Games and Interaction' in Amsterdam [February 2007]) have been dealing with logic in game and decision theory and dynamic logics with announcement and action operations. Fruitful technical advances have led to deep insights into the nature of communicative interaction by logicians.

This new direction of logic has quickly gained momentum and support. In 2006, a Marie Curie Research Training Site GLoRiClass ("Games in Logic Reaching Out for Classical Game Theory") was opened in Amsterdam, providing graduate student training for a large number of PhD students. In 2007, the European Science Foundation has recognized this direction as one of the foremost research developments for the European science community and created a collaborative research platform called "LogICCC – Modelling intelligent interaction". Later in 2007, a new book series entitled "*Texts in Logic and Games*" was launched by Amsterdam University Press.

While these interactive aspects are relatively new to logicians, on a rather different level, modelling intelligent interaction has been an aspect of the practical work of computer game designers for a long time. Pragmatic questions such as 'What makes a game/storyline interesting', 'What makes an reaction natural', and 'What role do emotions play in game decisions' have been tackled by practicing game programmers. The practical aspects of computer gaming reach out to a wide interdisciplinary field including psychology and cognitive science.

So far, there are only a few cross-links between these two communities. A number of logicians have applied logical methods for concrete games, such as van Ditmarsch's analysis of Cluedo [2], Sevenster's analysis of Scotland Yard [6, 7], and the new TACTICS project of van Benthem and van den Herik (represented at this symposium by the joint paper by Schadd, Winands, van den Herik, and Aldewereld).

Our symposium will explore the possibilities of joining the theoretical approach to interaction and communication with the practical approach to simulating behaviour. We would like to include purely logical aspects, cognitive and psychological aspects (including empirical testing of formal models), and pragmatic aspects.

2 A CASE FOR LOGIC

In § 1, we mentioned that questions such as 'What makes a game/storyline interesting', 'What makes an reaction natural', and 'What role do emotions play in game decisions' are relevant for game programmers and designers.

For more combinatorial games, such as strategic board games, the first question can be phrased as 'What technical properties of a game make it fun to play?'. In order to be enjoyable, a board game should neither be too complicated (as it would become frustrating) nor to simple (as it would become boring). A number of games that are played in practice have been found to be **NP**-complete[2]. Could it be that this technical notion from theoretical computer science is a good indicator for when a game is interesting?[3]

A different type of modelling can be found in interactive games, for instance the computer role playing games, in which the human player plays the role of some fictitious personality interacting with artificial agents, the so-called "non-player characters" (NPCs).

In these games, modelling interaction and behaviour becomes central and it is here that modern logic techniques such as dynamic logic, epistemic logic and variuous approaches to multi-agents systems could become useful.

[1] Institute for Logic, Language and Computation, Universiteit van Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands; `bloewe@science.uva.nl`; Department Mathematik, Universität Hamburg, Bundesstrasse 55, 20146 Hamburg, Germany; Mathematisches Institut, Rheinische Friedrich-Wilhelms-Universität Bonn, Beringstraße 1, 53115 Bonn, Germany.

[2] For instance, Minesweeper [3], Sudoku [8], Battleships [5], and Scotland Yard [6, 7]

[3] Personally, I think that this is rather unlikely, as **NP**-completeness is a property of a family of games parametrized by some number, typically the size of the game board, whereas the games that are actually played are always of one fixed parameter and thus oblivious to the fact that the family itself is **NP**-complete.

Logicians have developed logics in which we can formally reason about states of knowledge, states of belief, intentions, actions, consequences of our actions and combinations of all these. As an example consider the paper [4] by Eric Pacuit and the present author in which backwards induction techniques are used to analyse a typical TV soap opera storyline of a love triangle, deceit, false assumptions about other people, and disappointment in terms of a "game of changing and mistaken beliefs".

Typical applications of a logic of knowledge, belief, intention and action could be as follows, and we would very much like to see models for this being developed as a consequence of this symposium:

Scenario 1.

In a strategic computer game, the human player plays a developing population. Certain skills (including battle skills) are developed according to a skill tree (for instance, building a cannon can only be done after one of the artificial agents became a smith).

Modelling the intentions and actions of the opposing (computer-played) populations could be done by reasoning in some modal system that assesses the battle strength of the human-played population based on knowledge of their development stage. News of the sighting of a smith brought to the leader of the opposing population could be read as an increase in the likelihood that the human-played population has developed a cannon (and thus figure prominently in the reasoning of whether the computer-played population should attack or not).

Scenario 2.

In a computer role playing game, one could implement situations in which NPCs try to outmaneuver the human player. For instance, an NPC X might intend to kill NPC Y and gain some valuable object currently in the possession of Y.

Meeting the human-played agent, X wishes to find out how much the human-played agent knows about Y and what their relationship is. Based on this information, X would now try to trick the human-played agent into going and killing Y. This would require subtle communication skills of X, keeping in mind his own preferences and goals without giving them away. Such a communication could be modelled in a logic of knowledge, belief, agency and intention.

The scope of our symposium is wider than the two classes of games presented in this section (strategic board games and interactive computer games). Logic can play a role in all situations where interaction and behaviour are simulated, such as artificial pets (as in the paper *Theory and Practice of Social Reasoning: Experiences with the iCat* by Frank Dignum), analysis of human behaviour in game situations (as in the presentation *An experimental study of information mechanisms in the trust game: effects of observation and cheap talk* by Jürgen Bracht or the paper *Private Information and Inference about Inference* by Sobei Hidenori Oda, Gen Masumoto, and Hiroyasu Yoneda), automatised reasoning about diagrams (as in the paper *How can machines reason with diagrams?* by Mateja Jamnik) and others. The most accurate description of the scope of the symposium is the collection of presented papers that the reader can find in this volume.

3 THE SYMPOSIUM AND ITS STRUCTURE

Our symposium has a largely exploratory character: researchers from many different areas should get together to share the fundamental ideas and approaches of their respective fields. In order to get a proper representation of the fields involved, we decided to invite a number of speakers, generously funded by the Marie Curie Research Training Site GLoRiClass. Our invited speakers are Thomas Ågotnes (*Bergen, Norway*), Rafael Bordini (*Durham, England*), Frank Dignum (*Utrecht, The Netherlands*), Mateja Jamnik (*Cambridge, England*), and David Ethan Kennerly (*Los Angeles CA, United States of America*). We had invited two more speakers (Steffen Huck and Eric Pacuit) who had to cancel their trip for personal reasons. The registration fees, travel and accommodation expenses of the invited speakers were generously funded by the Marie Curie Research Training Site GLoRiClass.

Figure 1. Marie Curie Research Training Site GLoRiClass "*Games in Logic Reaching Out To Classical Game Theory*"

In addition to the invited speakers, the symposium attracted a large number of submissions from various communities (multi-agent systems, applied logic, experimental game theory, and others). All submissions (including the submissions of the invited speakers) were lightly refereed by the members of the programme committee and some external referees, keeping in mind the exploratory character of the symposium. We did not expect new research contributions, but interesting ideas for collaboration, and this is how the papers of the symposium have to be understood.

Programme Committee.

- Stefania Bandini, *Milan*
- Johan van Benthem, *Amsterdam* & *Stanford CA*
- Cristiano Castelfranchi, *Rome*
- Bruce Edmonds, *Manchester*
- Jaap van den Herik, *Maastricht*
- Wiebe van der Hoek, *Liverpool*
- Benedikt Löwe, *Amsterdam*
- Yoav Shoham, *Stanford CA*
- Keith Stenning, *Edinburgh*
- Rineke Verbrugge, *Groningen*

List of all presentations in alphabetic order.

- Thomas Ågotnes: *Logics of Interaction, Coalitions and Social Choice.*
- Rafael Bordini: *Simulating Rational Goal-Directed Behaviour Using a Logic-Based Programming Language for Multi-Agent Systems*
- Jürgen Bracht: *An experimental study of information mechanisms in the trust game: effects of observation and cheap talk*
- Jan Broersen: *Interpreting Product Update as Reasoning about Observations and Meta-Observations.*
- Flavio S Correa da Silva, Giuseppe Vizzari, Alessandro Mosca: *Coupled MMASS: A Formal Model for Non-deterministic Multi-agent Simulations*
- Louise Dennis, Bernd Farwer: *Gwendolen: A BDI Language for Verifiable Agents*
- Frank Dignum: *Theory and Practice of Social Reasoning: Experiences with the iCat*
- Mateja Jamnik. *How can machines reason with diagrams?*
- Ethan Kennerly: *Open Problems in Simulation and Story Analysis*
- Alessandro Mosca, Giuseppe Vizzari, Matteo Palmonari, Stefania Bandini: *A Perception Oriented MAS Model with Hybrid Commonsense Spatial Reasoning*
- Anton Nijholt: *Don't Give Yourself Away: Cooperative Behaviour Revisited.*
- Sobei Hidenori Oda, Gen Masumoto, Hiroyasu Yoneda: *Private Information and Inference about Inference*
- Maarten Schadd, Mark Winands, Jaap van den Herik, Huib Aldewereld: *Addressing* **NP**-*Complete Puzzles with Monte-Carlo Methods*
- Vincent Wiegel, Jan van den Berg: *Experimental Computational Philosophy: shedding new lights on (old) philosophical debates*
- Andreas Witzel, Jonathan A. Zvesper: *Higher-Order Knowledge in Computer Games*

ACKNOWLEDGEMENTS

The author would like to thank the organizers of the AISB 2008 Convention, Frank Guerin and Wamberto Vasconcelos for their support of this symposium by flexible and unbureaucratic decisions. The symposium is funded by the Marie Curie Research Training Site GLoRiClass (MEST-CT-2005-020841).

REFERENCES

[1] J. F. A. K. **van Benthem**, D. **Gabbay**, B. **Löwe** (*eds.*), Interactive Logic Selected Papers from the 7th Augustus de Morgan Workshop, London, Amsterdam University Press 2007 [Texts in Logic and Games 1]

[2] H. P. **van Ditmarsch**, The description of game actions in Cluedo, **Game theory and Applications** 8 (2002), pp. 1–28

[3] R. **Kaye**, Minesweeper is **NP**-complete, **Mathematical Intelligencer** 22 (2000), pp. 9–15

[4] B. **Löwe**, E. **Pacuit**, An abstract approach to reasoning about games with mistaken and changing beliefs, **ILLC Publication Series** PP-2006-33

[5] M. **Sevenster**, Battleships as decision problem, **ICGA Journal** 27 (2004), pp. 142-149

[6] M. **Sevenster**, Branches of imperfect information: logic, games, and computation, PhD Thesis, Universiteit van Amsterdam 2006 (**ILLC Publications** DS-2006-06)

[7] M. **Sevenster**, The Complexity of Scotland Yard, *in:* [1, pp. 209–246]

[8] T. **Yato**, Complexity and completeness of finding another solution and its application to puzzles, Master's Thesis, University of Tokyo, 2003

INVITED TALK
Logics of Interaction, Coalitions and Social Choice (extended abstract)

Thomas Ågotnes[1 2] and **Wiebe van der Hoek**[3] and **Michael Wooldridge**[4]

Abstract. While different forms of social interaction have been extensively studied in several fields, the development of formal logics makes precise knowledge representation and mechanical reasoning about situations involving social interaction possible. In particular, such logics make it possible to formally specify and verify software implementing social mechanisms. In my talk I will give an overview of some of our recent work on logics for social interaction, in particular applied to reasoning about social choice mechanisms such as voting and preference aggregation as well as reasoning about coalition formation and coalitional stability. We use benchmark examples from game theory and social choice theory to illustrate the expressiveness of the logics.

1 Introduction

Logics for reasoning about social interaction take into account the facts that individual agents are *autonomous*, *self interested*, and act *strategically* in order to obtain their goals. Such logics often try to give an account of what happens when agents choose to *cooperate* or otherwise act together [7, 17]. Of course, concepts such as strategic interaction, preferences, and cooperation have been extensively studied in fields such as game theory and social choice theory. However, the study of their *logical* and *computational* properties is relatively new.

Logics for social interaction are useful for, e.g., knowledge representation and reasoning in multi-agent systems [21] and for the formal specification and automated verification of *computational mechanisms* [19, 20, 13]. Given a specification of a mechanism in the form of a logical formula, we can use logical tools to *verify* (model checking) and/or *synthesise* (constructive proof of satisfiability) mechanisms [18, 23].

In this note we briefly discuss some new logics for social interaction. Many such logics have been suggested, and they differ, first and foremost, in their *expressiveness*. In order for a logic to be useful it must be sufficiently expressive, and we take concepts, properties and results from fields such as game theory and social choice theory as benchmark tests of expressiveness.

In this note we shall be particularly concerned with two aspects of interaction. The first is *social choice mechanisms*; a very general class of economic mechanisms [9] concerned with selecting some particular outcome from a range of alternatives on behalf of a collection of agents, such as *voting*, *preference aggregation* and *judgment aggregation*. The second is *coalition formation* in general and *coalitional stability* in particular. While it is clear that these are two distinct notions and problems, they are closely connected at least on one level of abstraction: they are both intimately related to the concept of *coalitional power* – what coalitions of agents can make come about. In the remainder of this note we very briefly and informally introduce some new logics for reasoning about these concepts, illustrated with example formulae (see the references for details). As a starting point we first mention Marc Pauly's seminal logic of coalitional power: Coalition Logic.

2 Coalition Logic

Two popular logics for social interaction are Alur, Henzinger and Kupferman's *Alternating-time Temporal Logic (*ATL*)* [7] and Pauly's *Coalition Logic (*CL*)* [17]. These logics let us express properties about the abilities, or powers, of single agents and of groups of agents acting together. The main syntactic construct of Coalition Logic is of the form[5]

$$\langle C \rangle \varphi,$$

where C is a set of agents, or a *coalition*, and φ is a formula. The intended meaning of $\langle C \rangle \varphi$ is that the agents C can choose to make φ come about, by performing some joint action. If $\langle C \rangle \varphi$ is true, then φ will not necessarily come about (C might choose to not make φ come about), but if $\langle C \rangle \varphi$ is true then C can guarantee that, no matter what the other agents do, φ will come about. ATL adds temporal operators such as "sometime in the future" to the language as well (CL can be seen as the next-time fragment of ATL).

Formally, formulae of Coalition Logic are interpreted on state-based structures where the agents play a strategic game, in the sense of non-cooperative game theory[6], in each state.

Pauly also observed [16] that the Coalition Logic construct can be used to express properties of social choice mechanisms. Consider the following example of a simple social choice mechanism [16]:

> *Two individuals, A and B, must choose between two outcomes, p and q. We want a mechanism that will allow them to choose which will satisfy the following requirements: We want an outcome to be possible – that is, we want the two agents to choose, collectively, either p or q. We do not want them to be able to*

[1] In my talk I will present joint work with Wiebe van der Hoek and Michael Wooldridge. In this extended abstract we give a brief outline.
[2] Bergen University College, Norway, email: tag@hib.no
[3] University of Liverpool, UK, email: wiebe@csc.liv.ac.uk
[4] University of Liverpool, UK, email: mjw@csc.liv.ac.uk

[5] Pauly [17] uses $[C]$ where we use $\langle C \rangle$.
[6] Strictly speaking, the structures of Coalition Logic associates a strategic game *form* to each state; a strategic game without preferences.

bring about both outcomes simultaneously. Finally, we do not want either agent to be able to unilaterally dictate an outcome – we want them both to have "equal power".

These requirements may be formally and naturally represented using CL, as follows:

$$\langle A, B\rangle p \quad (1)$$

$$\langle A, B\rangle q \quad (2)$$

$$\neg\langle A, B\rangle(p \wedge q) \quad (3)$$

$$\neg\langle A\rangle p \quad (4)$$

$$\neg\langle B\rangle p \quad (5)$$

$$\neg\langle A\rangle q \quad (6)$$

$$\neg\langle B\rangle q \quad (7)$$

Property (1) states that A and B can collectively choose p, while (2) states that they can choose q; (3) states that they cannot choose p and q simultaneously; and properties (4)–(7) state that neither agent can dictate an outcome.

3 Quantification

3.1 Quantified Coalition Logic

Expressing many interaction properties requires *quantification over agents and/or coalitions*. For example, consider the following *weak veto player* property [22]: "no coalition which does not have agent i as a member can make φ come about". This property can indeed be expressed in Coalition Logic as follows:

$$\bigwedge_{C \subseteq (Ag \setminus \{i\})} \neg\langle C\rangle\varphi$$

where Ag is the set of all agents in the system (the *grand coalition*). We thus use conjunction as a universal quantifier. The problem with this formulation is that it results in a formula that is exponentially long in the number of agents in the system. An obvious solution would be to extend CL with a first-order-style apparatus for quantifying over coalitions. In such a quantified CL, one might express the above by the following formula:

$$\forall C : ((C \subseteq Ag \setminus \{i\}) \rightarrow \neg\langle C\rangle\varphi)$$

However, adding quantification in such a naive way leads to undecidability over infinite domains (using basic quantificational set theory we can define arithmetic), and very high computational complexity even over finite domains. The question therefore arises whether we can add quantification to cooperation logics in such a way that we can express useful properties of cooperation in games *without* making the resulting logic too computationally complex to be of practical interest. In [2], we answered this question in the affirmative. We introduced *Quantified Coalition Logic (*QCL*)*, which allows a useful but restricted form of quantification over coalitions. In QCL, we replace cooperation modalities $\langle C\rangle$ with expressions $\langle P\rangle\phi$ and $[P]\phi$; here, P is a *predicate over coalitions*, and the two sentences express the facts that *there exists a coalition C satisfying property P such that C can achieve ϕ* and *all coalitions satisfying property P can achieve ϕ*, respectively. Examples of coalition predicates are, when C' is a coalition and n is a natural number:

- *supseteq*(C'): satisfied by a coalition C iff C is a superset of C'
- *geq*(n): satisfied by a coalition C iff C contains more than or equal to n agents
- *gt*(n): satisfied by a coalition C iff C contains more than n agents
- *maj*(n): satisfied by a coalition C iff C contains more than $n/2$ agents

For example, the property that agent i is a weak veto player for φ can be expressed as $\neg\langle\neg supseteq\{i\}\rangle\varphi$. Here the expression does not depend on the number of agents in the system. Thus we add a limited form of quantification *without* the apparatus of quantificational set theory. The resulting logic, QCL, is exponentially more succinct than the corresponding fragment of CL, while being computationally no worse with respect to the key problem of model checking.

To see how QCL makes it easier to express properties related to *voting*, consider the specification of *majority voting*:

An electorate of n voters wishes to select one of two outcomes ω_1 and ω_2. They want to use a simple majority voting protocol, so that outcome ω_i will be selected iff a majority of the n voters state a preference for it. No coalition of less than majority size should be able to select an outcome, and any *majority should be able to choose the outcome (i.e., the selection procedure is not influenced by the "names" of the agents in a coalition).*

Let $maj(n)$ be a predicate over coalitions that is satisfied if the coalition against which it is evaluated contains a majority of n agents. For example, if $n = 3$, then coalition $\{1, 3\}$ would satisfy the predicate, as would coalitions $\{2, 3\}$ and $\{1, 2\}$, but coalitions $\{1\}$, $\{2\}$, and $\{3\}$ would not. We can express the majority voting requirements above as follows. First: *every majority should be able to select an outcome*.

$$([maj(n)]\omega_1) \wedge ([maj(n)]\omega_2)$$

Second: no coalition that is not a majority can select an outcome.

$$(\neg\langle\neg maj(n)\rangle\omega_1) \wedge (\neg\langle\neg maj(n)\rangle\omega_2)$$

Simple though this example is, it is worth bearing in mind that its expression in CL is exponentially long in n.

3.2 Quantified Epistemic Logic

Epistemic logics [11, 14] give an account of agents' knowledge or beliefs. Operators K_i, C_G, E_G and D_G where i is an agent and G is a coalition are often used; $K_i\phi$, $C_G\phi$, $E_G\phi$ and $D_G\phi$ mean that i knows ϕ, that ϕ is common knowledge in the group G, that every member of G knows ϕ, and that ϕ is distributed knowledge in G, respectively.

The C_G, E_G and D_G operators let us express properties about group knowledge, but certain properties require quantification over agents and/or coalitions. Consider, for example, the following property:

At least two agents know that at most three agents know ϕ, from an overall set of agents $\{1, 2, 3, 4\}$.

A way to express this fact in conventional epistemic logic is as follows:

$$\begin{array}{l} E_{\{1,2\}}\psi \vee E_{\{1,3\}}\psi \vee E_{\{1,4\}}\psi \vee \\ E_{\{2,3\}}\psi \vee E_{\{2,4\}}\psi \vee E_{\{3,4\}}\psi \vee \\ E_{\{1,2,3\}}\psi \vee E_{\{1,2,4\}}\psi \vee E_{\{1,3,4\}}\psi \vee \\ E_{\{2,3,4\}}\psi \vee E_{\{1,2,3,4\}}\psi \end{array}$$

where ψ is:

$$(\neg K_1\varphi \vee \neg K_2\varphi \vee \neg K_3\varphi \vee \neg K_4\varphi)$$

The problem with this expression is similar to the problem with quantifying over coalitions in (standard) coalition logic discussed above: it is not very succinct, exponentially long in the number of agents in the system, and unrealistic for practical purposes. Again, we could add first-order style quantifiers, making it possible to express the property above as

$$\exists G : (|G| \geq 2) \wedge E_G \psi,$$

but this approach has the same disadvantages as discussed in the coalition logic case above.

But now we have a tool for limited quantification: coalition predicates. In [1] we introduce an epistemic logic with quantification over coalitions (ELQC), where the C_G, E_G and D_G operators are replaced by operators $\langle P \rangle_C$ and $[P]_C$, $\langle P \rangle_E$ and $[P]_E$, and $\langle P \rangle_D$ and $[P]_D$, respectively, where P is a coalition predicate. Now, $\langle P \rangle_C \phi$ means that *there exists a coalition G satisfying property P such that G have common knowledge of* ϕ, $[P]_C \phi$ means that *all coalitions G satisfying property P have common knowledge of* ϕ, and similarly for the two other kinds of group knowledge. The property discussed above can now be expressed as:

$$\langle geq(2) \rangle_E \neg \langle gt(3) \rangle_E \phi.$$

Possibly interesting properties of voting protocols include their knowledge dynamics. For example, when the winner of a voting protocol is announced, what does that tell an agent or a group of agents about the votes of other agents? ELQC can be used to reason about such properties. As an example, consider the following situation.

A committee consisting of Ann, Bill, Cath and Dave, vote for who should be the leader of the committee (it is possible to vote for oneself). The winner is decided by majority voting (majority means at least three votes, if there is no majority there is no winner).

Consider first the situation before the winner is announced. Let proposition a mean that Ann wins, and una_a that Ann wins unanimously, and similarly for the other three agents. The following ELQC formula holds (no matter what the actual votes are):

$$\neg a \rightarrow \langle geq(2) \rangle_D \neg \langle geq(3) \rangle_E (\neg una_b \wedge \neg una_c \wedge \neg una_d).$$

The formula says that if Ann does not win, there is a group of at least two agents who distributively know that at most two agents know that neither Bill nor Cath nor Dave wins unanimously.

Consider next the situation when, after the secret voting, the winner is announced to be Ann. Let *Vote* be a set of atomic formulae, each denoting a complete vote (e.g., "Ann, Bill and Cath voted for Ann, Dave voted for himself"). The ELQC formula

$$\bigwedge_{vote \in Votes} (vote \rightarrow [supseteq(\emptyset)]_C \langle gt(1) \rangle_E vote)$$

denotes the fact that no matter what the actual vote is, in any coalition it is common knowledge that at most one agent knows the actual (complete) vote. This formula is true after the winner is announced to be Ann.

4 Logics for Coalitional Games

As mentioned in Section 2, there is a strong connection between Coalition Logic and non-cooperative games. As a result of the inherent differences between the class of non-cooperative on the one hand and the class of *coalitional*, or *cooperative*, games (as studied in coalitional game theory [15, Part IV]), on the other, the usefulness of standard Coalition Logic in reasoning about the latter type of games is limited. One of the main questions related to coalitional games is: "Which coalitions will form?", or "Which coalitions are stable?". Solution concepts such as the *core* have been proposed in coalitional game theory in order to capture the idea of rational participation in a coalition. In [4, 3], we develop two logics, *Coalitional Game Logic* and *Modal Coalitional Game Logic*, for reasoning about such games. Both logics keep the main syntactic construct of Coalition Logic, but the formulae are now interpreted in the context of a (single) coalitional game.

A *coalitional game* (without transferable payoff) is an $(m + 3)$-tuple [15, p.268]: $\Gamma = \langle Ag, \Omega, \sqsupseteq_1, \ldots, \sqsupseteq_m, V \rangle$ where , $\sqsupseteq_i \subseteq \Omega \times \Omega$ is a complete, reflexive, and transitive *preference relation*, for each agent $i \in Ag$.

In [4, 3] we discuss these logics in detail, including axiomatisation, expressiveness and computational complexity.

4.1 Coalitional Game Logic

The main construct of Coalitional Game Logic (CGL) [4] is the CL construct $\langle C \rangle \varphi$, again with the intended meaning that C can make φ come about. In addition, CGL has symbols for referring to particular outcomes, as well as formulae of the form $\omega \succeq_i \omega'$, where ω and ω' are outcomes, meaning that agent i weakly prefers ω over ω'.

As an example of a coalitional game property expressed in CGL, take the following: "outcome ω is in the core". The *core* of a coalitional game is the set of outcomes which can be chosen by the grand coalition such that no coalition can choose a different outcome which is strictly better for all the agents in the coalition:

$$CM(\omega) \equiv \langle Ag \rangle \omega \wedge \neg \left[\bigvee_{C \subseteq Ag} \bigvee_{\omega' \in \Omega} (\langle C \rangle \omega') \wedge \bigwedge_{i \in C} (\omega' \succ_i \omega) \right]$$

expresses the fact that ω is a member of the core. The formula $CNE \equiv \bigvee_{\omega \in \Omega} CM(\omega)$ will then mean that the core is non-empty.

4.2 Modal Coalitional Game Logic

While the main construct of Modal Coalitional Game Logic (MCGL) [3] still is the familiar $\langle C \rangle \varphi$; its interpretation is here radically different: the intended meaning is that coalition *C prefers* φ. The formulae are now interpreted in the context of an outcome in a coalitional game, and $\langle C \rangle \varphi$ is true if there is some other outcome which is (weakly) preferred by every agent in C where φ is true. Similar modalities were used by Harrenstein [12] in the context of non-cooperative games. The operator $\langle C^s \rangle$ is used to denote *strict* preference in the same way, and the duals $[C]$ and $[C^s]$ denote that the formula is true in *all* outcomes preferred over the current outcome by all agents in C. In addition, the language has an atomic symbol p_C for each coalition C, meaning that the current outcome can be chosen by C.

The fact that the current outcome is in the core can now be expressed as:

$$MCM \equiv p_{Ag} \wedge \bigwedge_{C \subseteq Ag} [C^s] \neg p_C$$

Comparing CGL and MCGL, interpreted over games with a finite set of outcomes the former is more expressive than the latter. However, observe that the expression $CM(\omega)$ quantifies by taking a disjunction over all outcomes. When the set of outcomes are infinite, this property cannot be expressed in CGL. In contrast, MCGL can express solution concepts such as core membership (expressed by the property *MCM*) and non-emptiness of the core also for games with infinitely many outcomes. Furthermore, MCGL can express many properties more *succinctly* than CGL: observe the difference between $CM(\omega)$ and *MCM*.

5 Logics for Aggregation of Preferences and Judgments

Preference aggregation – the combination of individuals' preference relations over some set of alternatives into a single social preference relation – has been studied in social choice theory for quite a while. The following is an example of three individuals' preferences over three alternatives a, b, c:

1	$a > b$	$b > c$	$a > c$
2	$a > b$	$c > b$	$c > a$
3	$b > a$	$b > c$	$c > a$
PaMV	$a > b$	$b > c$	$c > a$

The example also shows the result of pair-wise majority voting (PaMV), and serves as an illustration of Condorcet's voting paradox: the result of PaMV is not always a proper preference relation (in the example it is cyclic). Arguably the most well known result in social choice theory is Arrow's theorem [8], saying that if there are more then two alternatives then no aggregation function can have all of a certain collection of reasonable properties (non-dictatorship, independence of irrelevant alternatives, Pareto optimality).

We argued above that CL can be used to express properties of social choice mechanisms. However, neither CL nor any of the other logics we have mentioned so far are expressive enough to be used for reasoning about certain important properties related to aggregation. A logic which can express such properties would be useful for, e.g., specifying and verifying electronic voting mechanisms.

Another link between aggregation and logic is the emerging field of *judgment aggregation* within social choice. Judgment aggregation is concerned with combining individuals' judgments on a set of logically interconnected propositions into a set of collective judgments on the same propositions. An example, illustrating voting in a committee on propositions "the candidate is qualified" (p), "if the candidate is qualified he will get an offer" ($p \to q$) and "the candidate will get an offer" (q) (Y[es]/N[o]):

	p	**p → q**	**q**
1	*Y*	*Y*	*Y*
2	*Y*	*N*	*N*
3	*N*	*Y*	*N*
PrMV	*Y*	*Y*	*N*

The example also shows the result of proposition-wise majority voting (PrMV), and serves as an illustration of the so-called *discursive dilemma*: although positions of the individual voters all are logically consistent, the result of PrMV is not. The similarity between Condorcet's paradox and the discursive dilemma suggests a relationship between classical Arrowian preference aggregation and judgment aggregation – and, indeed, recent research [10] shows that the former is a special case of the latter.

Judgment Aggregation Logic (JAL) [5, 6] was developed specifically for expressing properties about judgment aggregation mechanisms. In consequence, it can also be used for classical preference aggregation. We use the logic to study the relationship between preference aggregation and judgment aggregation. Being tailor made for the purpose, it is much more expressive than CL when it comes to aggregation. The logic can express, e.g.:

- aggregation rules such as pair-wise and proposition-wise majority voting;
- properties of aggregation mechanisms such as non-dictatorship, independence of irrelevant alternatives and Pareto optimality; and
- important results such as the discursive paradox, Arrow's theorem and Condorcet's paradox.

A sound and complete axiomatisation is provided. This effectively gives us a proof theory for social choice and judgment aggregation. For example, that we can express Arrow's theorem in the logic on the one hand, and that we have a sound and complete proof system on the other, mean that we have a formal way to prove Arrow's theorem. Thus, the logic might be useful not only as a tool for specifying and verifying computational mechanisms, but also as a computational tool for social choice.

We give a taste of JAL in the context of preference aggregation. The language has to pairs of dual modalities. $\Diamond$ ($\Box$) quantifies over preference profiles (one preference relation for each agent), and $\blacklozenge$ ($\blacksquare$) over pairs of alternatives; the diamonds denoting existential and the boxes universal quantification. In addition, there is an atomic formula i for each agent i, which is interpreted in the context of a preference profile and a pair $\langle a, b\rangle$ of alternatives, meaning that agent i prefers the first alternative (a) over the second (b). Finally, there is an atomic formula σ, interpreted in the context of an aggregation function, a preference profile and a pair of alternatives $\langle a, b\rangle$, meaning that in the result of aggregating the preferences the element a is preferred over b. Formulae can be seen as properties of aggregation functions. For example, the formula

$$ND = \bigwedge_{i \in \Sigma} \Diamond\blacklozenge\neg(\sigma \leftrightarrow i) \tag{8}$$

is the non-dictatorship property: for every agent i there is a preference profile and a pair of alternatives $\langle a, b\rangle$ such that it is not the case that both i and the aggregation prefers a over b. Another example is Pareto-optimality:

$$UNA = \Box\blacksquare((1 \wedge \cdots \wedge n) \to \sigma) \tag{9}$$

REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge, 'Quantifying over coalitions in epistemic logic', in *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, eds., Lin Padgham and David C. Parkes. IFAMAAS, (May 2008). To appear.

[2] Thomas Ågotnes, Wiebe van der Hoek, and Michael Wooldridge, 'Quantified coalition logic', in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, ed., M. M. Veloso, pp. 1181–1186, California, (2007). AAAI Press.

[3] Thomas Ågotnes, Wiebe van der Hoek, and Michael Wooldridge. Reasoning about coalitional games, 2008. Manuscript.

[4] Thomas Ågotnes, Michael Wooldridge, and Wiebe van der Hoek, 'On the logic of coalitional games', in *Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, eds., P. Stone and G. Weiss, pp. 153–160, Hakodate, Japan, (May 2006). ACM Press.

[5] Thomas Ågotnes, Michael Wooldridge, and Wiebe van der Hoek, 'Towards a logic of social welfare', in *Proceedings of The 7th Conference on Logic and the Foundations of Game and Decision Theory (LOFT)*, eds., Giacomo Bonanno, Wiebe van der Hoek, and Michael Wooldridge, pp. 1–10, (July 2006).

[6] Thomas Ågotnes, Michael Wooldridge, and Wiebe van der Hoek, 'Reasoning about judgment and preference aggregation', in *Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, eds., M. Huhns and O. Shehory, pp. 554–561. IFAMAAS, (May 2007).

[7] R. Alur, T. A. Henzinger, and O. Kupferman, 'Alternating-time temporal logic', *Journal of the ACM*, **49**(5), 672–713, (September 2002).

[8] K. J. Arrow, *Social Choice and Individual Values*, Wiley, 1951.

[9] *Handbook of Social Choice and Welfare Volume 1*, eds., K. J. Arrow, A. K. Sen, and K. Suzumura, Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 2002.

[10] Franz Dietrich and Christian List, 'Arrow's theorem in judgment aggregation', *Social Choice and Welfare*, (2006). Forthcoming.

[11] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, *Reasoning About Knowledge*, The MIT Press, Cambridge, Massachusetts, 1995.

[12] P. Harrenstein, *Logic in Conflict*, Ph.D. dissertation, Utrecht University, 2004.

[13] S. Kraus, *Strategic Negotiation in Multiagent Environments*, The MIT Press: Cambridge, MA, 2001.

[14] J.-J. Ch. Meyer and W. van der Hoek, *Epistemic Logic for AI and Computer Science*, Cambridge University Press: Cambridge, England, 1995.

[15] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, The MIT Press: Cambridge, MA, 1994.

[16] M. Pauly, *Logic for Social Software*, Ph.D. dissertation, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[17] M. Pauly, 'A modal logic for coalitional power in games', *Journal of Logic and Computation*, **12**(1), 149–166, (2002).

[18] M. Pauly and M. Wooldridge, 'Logic for mechanism design — a manifesto', in *Proceedings of the 2003 Workshop on Game Theory and Decision Theory in Agent Systems (GTDT-2003)*, Melbourne, Australia, (2003).

[19] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, The MIT Press: Cambridge, MA, 1994.

[20] T. Sandholm, 'Distributed rational decision making', in *Multiagent Systems*, ed., G. Weiß, 201–258, The MIT Press: Cambridge, MA, (1999).

[21] M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2002.

[22] M. Wooldridge and P. E. Dunne, 'On the computational complexity of qualitative coalitional games', *Artificial Intelligence*, **158**(1), 27–73, (2004).

[23] Michael Wooldridge, Thomas Ågotnes, Paul E. Dunne, and Wiebe van der Hoek, 'Logic for automated mechanism design – a progress report', in *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI 2007)*, ed., AAAI Press, Vancouver, Canada, (July 2007).

Simulating Rational Goal-Directed Behaviour Using a Logic-Based Programming Language for Multi-Agent Systems

Rafael H. Bordini[1]

Extended Abstract

An essential aspect of autonomous agents is that they must display *proactive* behaviour. Designing such software then requires explicit consideration of the *goals* the agent ought to achieve, and similarly its implementation also needs to be based on explicit representations of such goals. This is part of the reason why the BDI (Belief-Desire-Intention) agent architecture [16, 17] has since the early 90's been the best known architecture for developing software agents. As the BDI notions are also used in "folk psychology" (i.e., how people ordinarily refer to other people's behaviour) it also makes it useful for modelling goal-directed human behaviour [12].

As well as work on an agent architecture based on the BDI notions, much work was done to formalise these notions using modal logic [18]. Based on that work and also on practical implementation (as reactive planning systems) of the BDI architecture such as PRS [8], Rao created a simple, abstract agent programming language called AgentSpeak(L) [15]. Building on the basic constructs of logic programming, it provides an elegant language for the essential features of a BDI-based reactive planning system.

Starting from the initial definition of AgentSpeak, we have worked on various extensions of the language, for example to allow agents programmed in that language to communication using a speech-act based agent communication language [21]. We also did work to relate back the programming language constructs and the interpreter data structures to the modalities of BDI logic using the operational semantics of the language [5]. This is important for ongoing work on formal verification (more on this below).

While that work was mainly theoretical, it served as a basis for the development of a very elaborate, highly customisable platform for developing multi-agent systems called ***Jason***, which was done in joint work with Jomi Hübner, and made available *open source* at `http://jason.sf.net`. That work culminated in the recent publication of a textbook to make the ideas of agent programming using ***Jason*** accessible to wider audiences [4].

Various features of the ***Jason*** platform make it useful for modelling and simulation of social phenomena. Since the initial ideas of using agent-based techniques for modelling and simulating human societies [9] in the early 90's, this area has grown at incredible pace, with social scientists all over the world increasingly having interest in using computer simulation as a research method. However, most of the available tools for social simulation allow only very simple agents (with no cognition) to be used. Prominent researchers in that area have advocated the need for cognitive agents in certain advanced types of social simulation [6]. We are, therefore, in ongoing work [3, 1], incorporating features in ***Jason*** which will make it an even more powerful platform for developing software based on multi-agent systems techniques, but also facilitate its use as a tool for social simulation, in particular for modelling human goal-directed behaviour. Two examples are as follows:

Environments: In ***Jason***, environment models have to be programmed in Java. Needless to say, a more declarative, high-level language would be very useful for social simulation, where models of the environment are typically very important. This was the motivation that led to the development of the ELMS language, described in [13]. That work has recently been extended [14] to allow environment descriptions to have objects containing social norms that are to be observed only within the confines of an environment location, possible where an institution or organisation is situated (similarly to 'please refrain from smoking' or 'keep silence' signs). Another recent development [19] is the integration of ***Jason*** with a well-known approach for developing multi-agent environment based on the "artifact" abstraction [20], which could help in the development of very elaborate (distributed) environments.

Organisations: An important part of agent-oriented software engineering is related to agent *organisations*, which has received much research attention in the last few years. We are currently working on allowing specifications of agent organisations (with the related notions of roles, groups, relationships between groups, social norms, etc.) to be used in combination with ***Jason*** for the programming of individual agents. The particular organisational model we use is called $\mathcal{M}\text{OISE}^{+}$ [10]; an initial integration with ***Jason*** is discussed in [11], and available from `http://moise.sf.net`. One of the advantages of the approach is that the organisation specification is available for the agents to access and possibly change at run time.

An important use of logic-based techniques in the context of software development in multi-agent systems (in particular, with many of its application being "dependable systems") is for formal verification. In previous work, we devised model checking techniques for multi-agent systems programmed in AgentSpeak [2]. While in that work we were specifically interested in model checking multi-agent systems programmed in AgentSpeak, in a recent ongoing project, joint with Michael Fisher (see acknowledgements below) we are interested in developing techniques that would allow model checking for a variety of agent-oriented programming langauges [7].

[1] University of Durham, UK, email: r.bordini@durham.ac.uk

ACKNOWLEDGEMENTS

The work on model checking agent programming languages mentioned above is joint with Michael Fisher (University of Liverpool) and supported by EPSRC grant numbers EP/D054788 (Durham) and EP/D052548 (Liverpool).

REFERENCES

[1] Rafael H. Bordini, Antônio Carlos da Rocha Costa, Jomi F. Hübner, Álvaro F. Moreira, Fabio Y. Okuyama, and Renata Vieira, 'MAS-SOC: a social simulation platform based on agent-oriented programming', *Journal of Artificial Societies and Social Simulation*, **8**(3), (Jun 2005). JASSS Forum, <http://jasss.soc.surrey.ac.uk/8/3/7.html>.

[2] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge, 'Verifying multi-agent programs by model checking', *Journal of Autonomous Agents and Multi-Agent Systems*, **12**(2), 239–256, (Mar 2006).

[3] Rafael H. Bordini and Jomi F. Hübner, 'Agent-based simulation using bdi programming in *jason*', in *Agents, Simulation and Applications*, eds., Adelinde M. Uhrmacher and Danny Weyns, Taylor and Francis, (2008). To appear.

[4] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using **Jason***, Wiley Series in Agent Technology, John Wiley & Sons, 2007.

[5] Rafael H. Bordini and Álvaro F. Moreira, 'Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L)', *Annals of Mathematics and Artificial Intelligence*, **42**(1–3), 197–226, (September 2004). Special Issue on Computational Logic in Multi-Agent Systems.

[6] Cristiano Castelfranchi, 'The theory of social functions: Challenges for computational social science and multi-agent learning', *Cognitive Systems Research*, **2**(1), 5–38, (April 2001).

[7] Louise A. Dennis, Berndt Farwer, Rafael H. Bordini, Michael Fisher, and Michael Wooldridge, 'A common semantic basis for bdi languages', in *Fifth international Workshop on Programming Multi-Agent Systems (ProMAS-2007), held with AAMAS 2007, May 15th, Honolulu, HI*, eds., Mehdi Dastani, Amal El Fallah Seghrouchni, Alessandro Ricci, and Michael Winikoff, (2007). To appear in Springer's LNCS series.

[8] Michael P. Georgeff and A. L. Lansky, 'Reactive reasoning and planning', in *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI'87), 13–17 July, 1987, Seattle, WA*, pp. 677–682, Manlo Park, CA, (1987). AAAI Press / MIT Press.

[9] *Simulating Society: The Computer Simulation of Social Phenomena*, eds., Nigel Gilbert and Jim Doran, UCL Press, London, 1994.

[10] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier, 'Using the $\mathcal{M}$OISE$^+$ for a cooperative framework of MAS reorganisation', in *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, eds., Ana L. C. Bazzan and Sofiane Labidi, volume 3171 of *LNAI*, pp. 506–515, Berlin, (2004). Springer.

[11] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier, 'Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels', *International Journal of Agent-Oriented Software Engineering*, **1**(3/4), 370–395, (2007).

[12] Emma Norling, 'Folk psychology for human modelling: Extending the bdi paradigm', in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 August 2004, New York, NY, USA*, eds., Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, pp. 202–209. IEEE Computer Society, (2004).

[13] Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa, 'ELMS: an environment description language for multi-agent simulations', in *Environments for Multiagent Systems, State-of-the-art and Research Challenges. Proceedings of the First International Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS-04, 19th of July*, eds., Danny Weyns, H. van Dyke Parunak, Fabien Michel, Tom Holvoet, and Jacques Ferber, number 3374 in Lecture Notes in Artificial Intelligence, pp. 91–108, Berlin, (2005). Springer-Verlag.

[14] Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa, 'Spatially distributed normative objects', in *Proceedings of the Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN), held with ECAI 2006, 28th August, Riva del Garda, Italy*, eds., Guido Boella, Olivier Boissier, Eric Matson, and Javier Vázquez-Salceda, (2006).

[15] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 22–25 January, Eindhoven, The Netherlands*, eds., Walter Van de Velde and John Perram, number 1038 in Lecture Notes in Artificial Intelligence, pp. 42–55, London, (1996). Springer-Verlag.

[16] Anand S. Rao and Michael P. Georgeff, 'Modeling rational agents within a BDI-architecture', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, eds., James Allen, Richard Fikes, and Erik Sandewall, pp. 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, (1991).

[17] Anand S. Rao and Michael P. Georgeff, 'BDI agents: From theory to practice', in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), 12–14 June, San Francisco, CA*, eds., Victor Lesser and Les Gasser, pp. 312–319, Menlo Park, CA, (1995). AAAI Press / MIT Press.

[18] Anand S. Rao and Michael P. Georgeff, 'Decision procedures for BDI logics', *Journal of Logic and Computation*, **8**(3), 293–343, (1998).

[19] Alessandro Ricci, Michele Piunti, L. Daghan Acay, Rafael H. Bordini, Jomi F. Hübner, and Mehdi Dastani, 'Integrating heterogeneous agent-programming platforms within artifact-based environments', in *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), 12–16 May, Estoril, Portugal*, eds., Lin Padgham, David Parkes, Jörg Müller, and Simon Parsons. International Foundation for Autonomous Agents and Multiagent Systems (`www.ifaamas.org`), (2008). To appear.

[20] Alessandro Ricci, Mirko Viroli, and Andrea Omicini, "'Give agents their artifacts": The A&A approach for engineering working environments in MAS', in *6th International Joint Conference "Autonomous Agents & Multi-Agent Systems" (AAMAS 2007)*, eds., Edmund Durfee, Makoto Yokoo, Michael Huhns, and Onn Shehory, pp. 601–603, Honolulu, Hawai'i, USA, (14–18 May 2007). IFAAMAS.

[21] Renata Vieira, Alvaro Moreira, Michael Wooldridge, and Rafael H. Bordini, 'On the formal semantics of speech-act based communication in an agent-oriented programming language', *Journal of Artificial Intelligence Research*, **29**, 221–267, (June 2007).

Interpreting Product Update as Reasoning about Observations and Meta-Observations

Jan Broersen[1]

Abstract. In this brief note, I would like to suggest that it makes sense to reinterpret product update, as introduced by Baltag, Moss and Solecki, as a system to account for observations and meta-observations, where a meta-observation is an observation of an observation. Under this interpretation we also take products of action models with meta-action models. I deliberate on some possible consequences of this extension to the interpretation of product update.

1 Introduction

Product update, as defined by Baltag, Moss and Solecki [1, 2], is about updating multi-agent epistemic models by modeling the assimilation of new information as a (restricted) modal product with a multi-agent epistemic action model. This paper reports two observations[2] concerning this framework. The first observation is that the mechanism defined by taking products only fits with an interpretation of the actions in action models as *observations*. The second, related observation is that action models themselves might be viewed as resulting from products of meta-action models representing meta-observations. These (admittedly preliminary) ideas may give rise to new action languages for talking about epistemic action models.

2 Product updates model observations

That the possible worlds resulting from a product update are a Cartesian product of the original worlds and the actions, is intuitively explained by the observation that in principle any of the epistemic actions in the action model can be performed from any state in the static epistemic model. We get a *restricted* product by considering the preconditions of the actions that prevent some actions to be performed from certain states. For the uncertainty relation in the product models the intersection of the uncertainty relations of the epistemic model and the action model is taken.

Surprisingly, in the literature not much effort is spend on explaining why it is that we have to take the intersection of the uncertainty relations originating from the static epistemic model and the epistemic action model. Baltag and Moss [2], in their most recent account of product update, say the following:

> ”We model the update of a state by an action as a partial update operation, given by a restricted product of the two structures: the uncertainties present in the given state and the given action are multiplied, while the impossible combinations of states and actions are eliminated (by testing the actions preconditions on the state). The underlying intuition is that, since the agents uncertainties concerning the state and the ones concerning the simple action are mutually independent, the two uncertainties must be multiplied, except that we insist on applying an action only to those states which satisfy its precondition.”

The quote explains that the intersection reflects multiplication of *independent* uncertainties. But the quote does not explain why we can assume this independency, nor does it explain what kind of actions actually ensure independency under the constraints imposed by the preconditions.

I will approach the question concerning the reason for taking an intersection from a slightly different angle. Prima facie, one might consider taking an intersection surprising: if an agent performs the same action from states he cannot distinguish, it will also not be able to distinguish the result states. And if an agent does not distinguish two actions from a state it does distinguish, again two or more indistinguishable effect states will result. This would then rather suggest that we should take the *union* instead of the intersection. So why is it that the intersection is the right choice? The rough answer is: because the actions of action models are ‘testing’ or ‘observation’ actions. Such actions always aim at *reducing* uncertainty. Furthermore, what these actions observe, must be true in the state where their precondition holds. So there is just exactly only *one* way in which observation actions can result in uncertainty: from an uncertain state it must be uncertain whether the observation has taken place. That explains the intersection.

This also sheds light on the question above, concerning the reason for the independence of the uncertainties involved. The independence is explained by the reasonable assumption that observations themselves do not interact with the conditions observed[3].

The term ‘observation’ should not be taken too literally here. Actually, from the level of abstraction we are looking at information transfer, ‘observation’, ‘testing’, ‘learning’ and ‘announcing’ are all alike. The difference between these concepts can only become more clear if we can differentiate between sending and receiving agents, their motivations for transferring information, and their strategies for dealing with new information. The present setting, without notions like ‘agency’ or ‘intention’ is simply too abstract for that.

3 Product update and meta-observation

Many standard examples are explicitly about observations. A well-known one is the following [3, p.130]:

Example 1 *In a situation with three colored cards and three agents*

[1] University of Utrecht, The Netherlands, email: broersen@cs.uu.nl
[2] As far as I know, these observations have not been reported before.
[3] Note that this assumption conflicts with the uncertainty principle from quantum physics.

knowing their own card, agent 1 publicly observes a card of agent 3 (for instance, because 3 shows it to him).

The action model distinguishes three different observations: '1 observes 3 has red' (1 : $r@3$), '1 observes 3 has white' (1 : $w@3$) and '1 observes 3 has blue' (1 : $b@3$). Agent 1 and 3 distinguish these actions, agent 2 does not. Below are the pictures for the static initial epistemic model, the action model, and the product model.

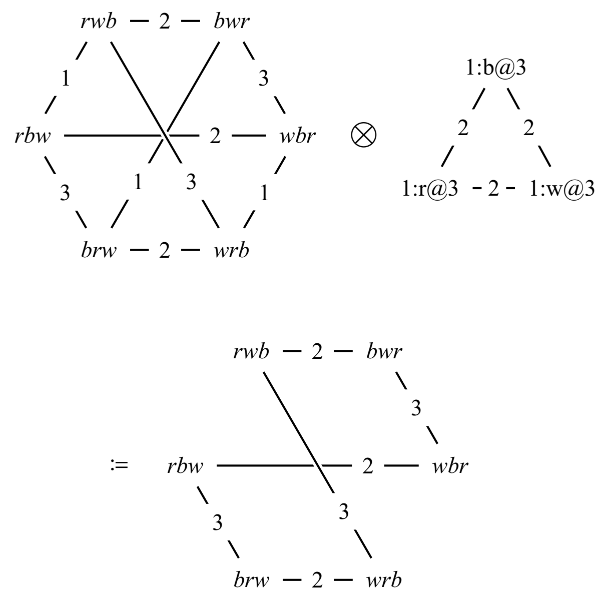


Figure 1. Agent 1 publicly observes agent 3's card

Now note that the description of example 1 also says that the observation itself is publicly observed. This is the first sign that something like 'meta'-observations are relevant for the example. In the following we will suggest that these meta-observations can be modeled as action models themselves. We will refer to these models as 'meta-action models'.

But now let us first extend the above example in order to make more clear what we mean.

Example 2 *Agent 3 has left the table, leaving his card on the table. After coming back he suspects that 1 has taken a look at his card, which, in fact is indeed what happened, and it happened publicly. Agent 3 publicly announces his suspicion.*

Figure 2 gives the right product model resulting from taking the product with the appropriate action model for this example. The model contains both the epistemic model of the initial situation and the epistemic model resulting from the previous example, and agent 3 hesitates between these two models. But what is the action model that produces this product model? Of course, it is not too difficult to find the right action model. However, below we show we can decompose this problem into two problems: finding the appropriate action model and finding the appropriate meta-action model.

It is rather clear that in this example there are at least two levels of observation. First there is the level where individual agents get to know the results of the card showing action. This level is exactly

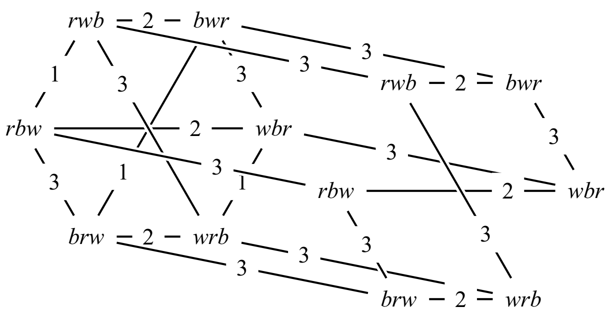


Figure 2. Agent 3 suspects agent 1 has seen his card

the same as in the first example. Therefore, in figure 3, that gives the action models for the observation and the meta-observation levels, the observation level action model is the same as the one in figure 1 for the first example (with the exception of the non-connected 'skip' world, which we discuss later). Indeed, we might say that the only difference between the two examples is on the level of meta-observations: in the first the meta-observation is like a public announcement, and in the second the meta-observation action model contains at least two meta-observation actions '3 observes that 1 takes a look at 3's cards' and '3 observes that 1 observes nothing'. Agent 3 does not distinguish between these actions (because he does not know whether the looking action actually took place). But agents 1 and 2 can distinguish between the two. Note that this meta-level action model models the suspicion of agent 3 as the hesitation about whether or not some observation has taken place on the observation level. Maybe the hesitation and suspicion originates from agent 3 not being sure whether or not he saw that agent 1 was taking a look at his card. Also note that the meta-level contains a third action: the meta-action of agent 3 not meta-observing anything at all ('skip'). To make the view using meta-levels of observation work, for all agents in the system we have to add such 'non-observation/skip' actions at any level and meta-level. Note that in the meta-level action model of figure 3, I only give the non-action for agent 3. Actually, to prevent the picture from being flooded by states that are inessential for explaining the idea I do not give any of the meta-level observation actions of agents other than agent 3. This is why the figure says 'etc.' in the meta-level action model. In particular, as long as at the direct meta-level there is no uncertainty about inaction, the non-actions can be neglected. For instance, note that using the observation action model of figure 3 in stead of the one in figure 1 to solve the first example, does not make a difference. In particular, if we stick to the original set-up, with only one level of action models, adding a non-observation action to the action model does not make a difference as long as there are no uncertainty relations linking it to other actions. Finally, note that the product model resulting from the product of the observation model and the meta-observation model, when multiplied with the static epistemic model, yields the product model of figure 2, as required.

Now, what are the preconditions for the actions in this meta-observation action model? And how exactly do we define the product of the meta-observation model and the observation model? The preconditions for the meta-observations are straightforward. Just like for the observation action models they just say that we can only observe what is actually true. For the action 'observing nothing' this

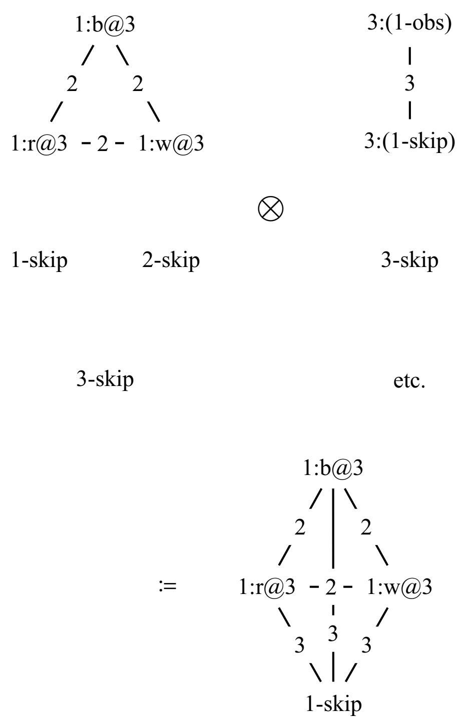


Figure 3. The meta-observation as a meta-product

means we get $\top$ as a precondition, because it is an action that can always be done. And as expected, since this is all about observation, for the product of an action model and a meta-action model we also take the intersection of uncertainty relations. Finally, we get the 'restricted' product by checking the preconditions of the meta-action model on the action model. In the specific example we deal with here, we have three meta-observations. The meta-observations $3:(1-\text{obs})$, $3:(1-\text{skip})$ and $3-\text{skip}$. The first meta-action has a precondition that is true on all actions of the observation model where agent 1 observes something. The second has a precondition that is true only on the action in the action model where agent 1 observes nothing. Finally, the precondition of $3-\text{skip}$ is true on all actions in the actions model. But this part of the meta-product only yields an unconnected copy of the action model.

In example 2 above, there is also a third meta-level of observation. Because of the involvement of this level, we say that agent 3 announces its suspicion. This ensures that the third meta-level is a public announcement. Actually, for any example we come up with, in principle infinitely many layers of observation play a role. Any observation is itself an action for which we can ask whether it is observed. So, for any observation potentially infinitely many actions are performed at the same time: every observation possibly has a meta-observation. In the examples above, this infinite hierarchy is broken down by the first meta-level of observation that is a public announcement. Actually, for any well-specified example, the highest relevant meta-level of observation is always a public announcement. If not, the example is under-specified, and leaves room for a non-deterministic interpretation of the update. Actually, in most examples in the literature, implicitly a closure assumption is applied: if nothing is said about the meta-levels of observation, it is assumed that they are public observations closed under meta-observations.

4 Future Research

The setting raises several questions. I briefly mention a few of them. The first is that products and meta-products are *not associative*. This is quite easy to see from the example above. The meta-products should always be taken first. In particular, if we first take the product of the static model and the observation level action model, it is not clear how to take the product of the resulting product model with the meta-level action model. Performing products in this order is not even well-defined. But also it is clear that we throw away information by first taking the product with the action model instead of the product with the meta-action model. We cannot recover this information. A possible problem is that associativity might be important for certain examples. For instance, what if only at a later moment an agent learns that his looking at the cards was observed or suspected by another agent. Since we do note have associativity, this can only be modeled by keeping track of all action and meta-action models over time.

Another interesting question is how we can add *agency* to the picture. Actually, viewing the actions as observations of specific agents, as we did in the above examples, is a first step in this direction. For every observation action it is important to describe whose observation it is, since on the meta-observation level this information is used. And for each observation action it is important to describe whose action is observed. Above we used an ad-hoc notation to describe actions. An obvious route for investigation is to try to turn this notation into an action language for observations and meta-observations.

One of the principles that suggest themselves is that agents always observe their own observations. Note that the setting we sketched actually does enable us to model situations where this is not the case. However, methodologically it would be strange to allow this. We do not want to get into the territory where agents make 'sub-conscious' observations.

The current set-up also enables us to speculate about a view where *all* knowledge is viewed as observation[4], even at the 'lowest' level. It is not too difficult to translate a standard static epistemic model into an action model containing the same information. This is accomplished by also seeing the knowledge of each agent at the lowest level as an observation. For instance, in the cards example, the static model is equivalent to the action model where each agent observes his own card.

I want to finish with a comment. It seems not right to claim that the setting we sketched adds nothing new only *because* the meta-products will simply return an action model of the form already available in the original setting of Baltag, Moss and Solecki. If that would be a good argument against the present proposal, the original proposal could be attacked with the same argument: product update adds nothing new, because it can be seen as a system that only specifies a complicated way to describe standard epistemic models.

REFERENCES

[1] L. S. Moss A. Baltag and S. Solecki, 'The logic of public announcements and common knowledge', in *Proceedings of TARK 1998*, (1998).
[2] A. Baltag and L. S. Moss, 'Logics for epistemic programs', *Synthese*, **139**, 165–224, (2004).
[3] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi, *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*, Springer, New York, NY, USA, 2007.

[4] Like in logical positivism.

Coupled MMASS: A Formal Model for Non-deterministic Multi-agent Simulations

Flavio Soares Correa da Silva[1] and **Giuseppe Vizzari** and **Alessandro Mosca**[2]

Abstract. The Multilayered Multi-agent Situated System (MMASS) has been proposed as a general framework to build multi-agent systems in which agents are situated in an environment whose characterization can be multifaceted (each facet of the environment is named a *layer*, hence the name of this framework). Agents in the MMASS can be purely reactive and computationally lightweight, or deliberative and employ highly sophisticated reasoning mechanisms. As a consequence, the MMASS has proven to be useful to build massively multi-agent systems (in which typically each agent is computationally simple) as well as systems comprised by complex agents (in which typically we have few agents interacting with each other and with the environment). In the present article we combine a simplified version of MMASS with a specific logical system, which we suggest that can be particularly suitable to solve problems based on simulations. The proposed logical system is a variation of classical FOPL, in which logical statements are tagged with probability values so that one can reason *probabilistically* as opposed of reasoning *with* probabilities or reasoning *about* probabilities.

1 INTRODUCTION

The Multilayered Multi-agent Situated System (MMASS) has been proposed as a general framework to build multi-agent systems in which agents are *situated* in an environment whose characterization can be multifaceted (or, using the MMASS parlance, *multilayered*). In this system, agents interact with each other and with the environment by:

- sensing and emitting fields across the different layers of the environment;
- communicating directly with each other;
- acting on the environment e.g. by picking and dropping objects; and
- moving about [11].

A distinguishing feature of the MMASS framework is that it accommodates a wide range of agent architectures, from computationally lightweight, purely reactive agents to highly demanding, sophisticated deliberative agents. The flexibility of the MMASS framework makes it useful to build multi-agent systems comprised by heterogeneous agents ranging from complex deliberative architectures to simplified and computationally tractable ones.

The MMASS framework has proven to be particularly useful to model massively multi-agent systems, which can be used to analyze system configurations based on empirical simulations (see, e.g., [3, 2]).

The flexibility of the MMASS framework stems from its agent architecture agnosticism: the MMASS framework is not related to any particular agent architecture, and this is how it can accommodate a wide range of different agent architectures. The down side of it is that, in order to employ the MMASS framework for practical problem solving, one needs to complement it with suitable agent architectures.

In the present work we introduce a specific architecture for agents to be integrated with the MMASS framework, which is quite general and seems of particular interest to model complex systems whose analysis must be based on simulations.

Our proposed architecture is paired with a slightly simplified version of the original MMASS framework. It is a variation of classical FOPL in which logical statements are tagged with probability values. As a motivating example, if we have that $\alpha \rightarrow \beta$ and that α is tagged with a probability value of 0.2, then we will be able to infer β whenever α is observed, i.e. our logical system will sometimes infer β (when α is true) and will some other times *not* infer β (when α is false), since α is a premise to infer β and it is sometimes true (with probability 0.2) and sometimes false (with probability 0.8). In the long run, and assuming that no other proof path can infer β or influence on the probability of α, β will be inferred as true about twenty percent of the times we attempt to infer it.

Notice that our proposed non-deterministic logical system is appropriate for, literally speaking, *uncertain reasoning*, as opposed to reasoning *with* uncertainty or reasoning *about* uncertainty, i.e. we are not designing a system to reason about probability values, nor we are designing a system whose inferences are subject to probabilistic truth valuations [5]. Rather, we are building a class of logical theories, from which a specific theory is built dynamically based on probabilistic selections. In other words, we build a collection of logical statements and attach to each of them a probability value, and then we pick a collection of statements, which are drawn according to their probability values, and assemble a logical theory that is used to perform an inference. If the same inference is tried again, a new set of statements is drawn, and therefore the result of this inference may change.

Systems for uncertain reasoning can be more useful for simulation based analysis of problem solving mechanisms (and randomness is also a recognized element in dealing with the variability of human behaviours in simulation [4]), contrasting with systems to reason about uncertainty that can be more useful for theoretical analysis of problem solving mechanisms and systems.

This paper is organized as follows: in section 2 we briefly introduce the MMASS framework and compare it with other similar

[1] University of Sao Paulo, BRAZIL, email: fcs@ime.usp.br
[2] University of Milano-Bicocca, ITALY, email: { giuseppe.vizzari, alessandro.mosca }@disco.unimib.it

work found in the literature. In section 3 we simplify the original MMASS model and put this simplified version together with our proposed non-deterministic system for deliberative agents. In section 4 we build a simple illustration, to show our proposed model in action. Finally, in section 5 we present some discussion, conclusions and proposed future work.

2 THE MMASS FRAMEWORK

The MMASS framework was fully introduced in [11], although some partial renditions of this framework had been published before that. It has been employed successfully in a variety of problems, such as the modeling of human interactions in Computer Supported Cooperative Work [8], simulation of crowd behavior [1] and ubiquitous computing [7].

Very briefly, a MMASS is a collection of *layers*. Each layer is comprised by an undirected graph, in which nodes represent locations in which agents can be placed (each node admits at most one agent) and edges represent accessibility between nodes. Agents can move across nodes following their accessibility relations, and in the original MMASS model agents can only exchange messages with other agents located in neighboring nodes.

In the original MMASS model, there exist *interfaces* connecting different layers. Essentially, an interface is an edge connecting nodes from two different graphs.

An agent, when located in a node, can sense the environment and perceive fields that are reaching that node, receive messages from neighboring agents, and then, based on internal states and deliberative capabilities, decide to move to a different node, act upon the environment, emit fields and send messages to agents.

A *field* is characterized by its source, initial intensity, diffusion and dispersion. A field source is the node from which it is being emitted; the initial intensity is typically represented as a non-negative real number; diffusion determines the intensity of the field as a function of the topological distance between any node and the field source; and dispersion determines the intensity of the field at the source node as a function of the time since the field was emitted.

The power of the MMASS framework stems from its flexibility, which is a direct consequence of its simplicity. In order to be applied in concrete situations, it must be complemented by specific agent models that determine agent behaviors based on sensed fields and received messages. Frequently, the specification of agent behaviors comes together with some sort of specialization of the generic MMASS framework, to build special cases of interest to particular applications.

In the present work we propose a partial specialization of the MMASS framework, thus building a special case of the generic framework that can be particularly well suited to model complex multi-agent systems whose dynamics must be analyzed based on empirical discrete-event simulations.

Our proposed partial specialization of the generic MMASS framework is built in order to couple it with a particular non-deterministic inference system, as detailed in the following section.

3 COUPLING MMASS WITH A NON-DETERMINISTIC LOGICAL SYSTEM

Our goal is to put the MMASS framework together with a specific agent model, to build a specialized framework for simulation of complex multi-agent systems. Essentially, we have built a normative theory based on which agents can infer obligations and permissions, i.e. logical statements representing actions that they are either required to perform (obligations) or allowed to perform (permissions). We intend this normative theory to work as a foundation upon which concrete theories of action for agents can be built. We also intend that the normative theory, heretofore referred to as *Coupled MMASS*, can be implemented to support runs of an agent system, i.e. concrete simulations that can be used to analyze empirically the behavior of complex systems.

Permissions and obligations connect naturally to *deontic logics* [9], a particular class of multimodal logics that have been historically used to model legal systems, and more recently have been employed to model interactions in multi-agent systems [10]. We have explored deontic logics as a formalism to model multi-agent systems for digital entertainment, and obtained rather appealing results [6].

For the purpose of the simulation of the dynamics of interactions among agents in a multi-agent system, however, deontic logics may not be the most appropriate formalism, as proof systems for even relatively simple deontic logical systems can become computationally intractable. For this reason, we have preserved the *notion* of permissions and obligations and explored simpler logical systems, that can be less computationally demanding and therefore more useful for simulations of practical systems.

We have extended the language of classical FOPL with *deontic tags* as follows. We have added to the alphabet of FOPL the tags **O** (standing for *obligation*) and **P:**φ (standing for *permission*), in which φ is a real number in the interval $[0, 1]$.

Classical FOPL sentences can be tagged with zero or one deontic tags. Operationally, when an **O** tag is found, the corresponding sentence is forcefully true, i.e. **O** tags can be ignored in the course of a proof. When a **P** tag is found, a random number generator is trigged to generate a random value in the interval $[0, 1]$ based on a uniform probability distribution. If the generated value is larger than φ, the tagged sentence is evaluated as false, otherwise it is true.

Intuitively, obligations are necessarily true, and permissions can be either true or false, according to a probability estimate.

Using this logical system, well formed sentences are sometimes labeled as theorems and some other times labeled as non-theorems for the same logical theory. Logical proofs are non-deterministic, depending on the interdependence of well formed sentences and the behavior of permitted sentences, which is determined by the values φ.

We have envisaged this non-deterministic formal system to model the actions of and interactions among agents in multi-agent systems. We now couple this system with MMASS.

Before putting the non-deterministic FOPL extended with permissions and obligations and MMASS, we simplify the original MMASS in three ways, to make it more specifically applicable for simulation models.

1. We add an explicit and fixed notion of *time*, based on the simplest possible model of time, namely discrete linear time with one starting point. Since we aim at simulations and animations, we add a uniform and very simple representation of time in the model. Time is represented as clock ticks, which are synchronized for all entities and layers, i.e. we have a universal clock accessible to any element of the model. A clock tick is a natural number. The original MMASS accommodates more complex representations of time, hence by reducing the set of possible representations to this single possibility we do not need to consider other alternatives, and Coupled MMASS is significantly simplified.
2. We enforce that all layers have the same topological structure. In

other words, we have a single *layer structure* i.e. an undirected graph - that is applied to all layers in an MMASS model. The edges of the layer structure are tagged with real, non-negative values, with the addition of the symbol $\perp$ that are representations of costs to move between nodes and for fields to propagate. The tags can vary across layers.
Edge tags are abstractions of some appropriate notion of distance. For example, one of the layers can represent geometrical distances between nodes, and tags can represent these distances using some uniform distance unit measure. This is the reason why we assume that the tags are non-negative real numbers. Notice that we leave on purpose the possibility of having zero as a legal tag. This can be used to collapse nodes in a layer without having to change the layer topology. The symbol $\perp$ represents instead an infinite distance, meaning that agents cannot move from one of the connected sites to the other, and also fields do not diffuse across the edge.

3. We assume that one layer represents the physical layer, on which agents can move, and that the movements of an agent imply on travel costs that are calculated independently for each layer. Agents move across the layer topology, and therefore
   (a) When an agent moves from one node to another, it does so in *all* layers; and
   (b) It makes no sense for an agent to migrate between layers, since agents do not move from the physical layer. Therefore, we do not need to have explicit interfaces between layers.

This basic structure for agents' environment supports the representation of composite situations, characterized by different aspects that may influence agents' behaviours. In particular, Figure 1 depicts a three–layered environment comprising (i) a basic grid whose edges represent airline costs of movement between sites, (ii) an additional layer in which edges represent the actual possibility to move from a site to another (e.g. a map of a building including walkable space, passages and walls) that could be used to effectively constraint agents' movements, but also (iii) a layer connecting "homogeneous" sites (e.g. indoor or outdoor sites) with edges of null cost, creating areas in which fields diffuse uniformly, for instance to support agents in distinguishing the different areas.

For each layer we also have the definition of a finite collection of *fields*. A field is determined by four elements:

1. A source node p_0, i.e. the identification of a node in the layer topology;
2. A starting moment t_0, i.e. a natural number;
3. An initial intensity f_0, i.e. a real, non-negative number;
4. A field equation, i.e. any mathematical relation that univocally determines the intensity f of the field at any node p and at any moment $t \geq t_0$, based on the topological distance between p and p_0. As a simplifying assumption for Coupled MMASS, we assume that fields are represented as real non-negative values that depend primarily on time interval and topological distance between nodes.

Fields and edge tags are the available resources in this framework to model *context*.

We now add entities to our model. Entities can be of two basic sorts:

1. Objects: an object is any inanimate entity, i.e. an entity that does not perform actions, does not emit fields and does not move by itself. We can have an arbitrary number of objects sitting on any node.

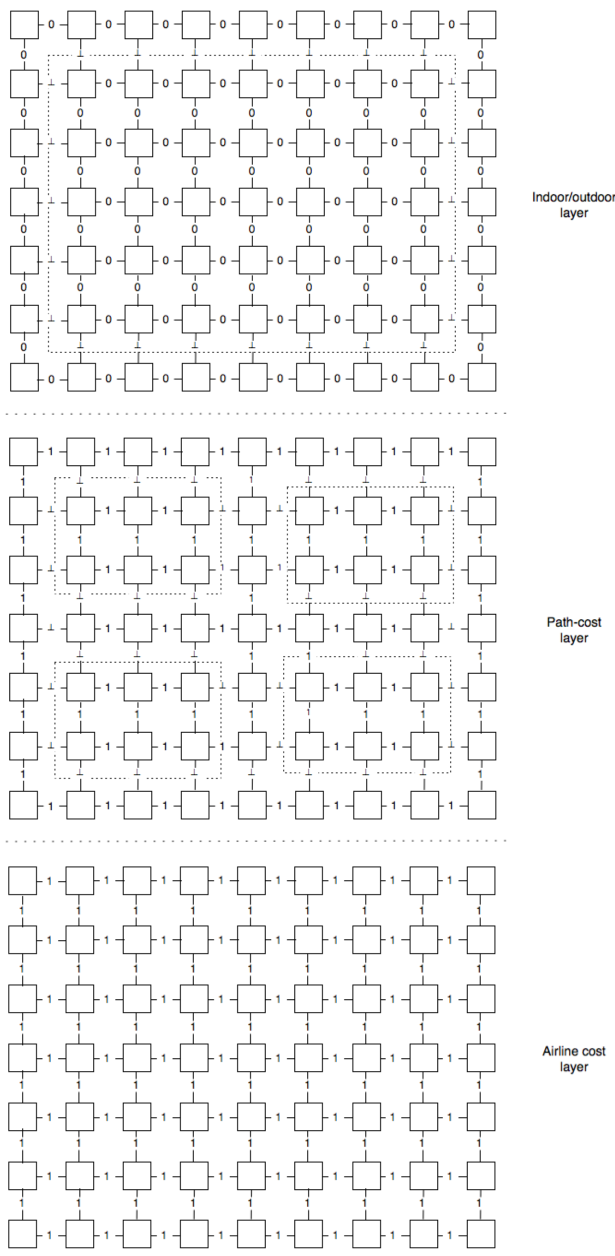


Figure 1. A sample multilayered structure of a coupled MMASS environment.

2. Agents: an agent is an entity that performs actions including actions that interact with objects emits fields and moves across the edges of the layer topology. We can have at most one agent on a node at any given moment.

An agent can tune to a layer and sense the fields that are hitting it at the node where it is. Fields are composed based on specified composition laws. Fields interact only with other fields at the same layer. An agent can also communicate with other agents. We simplify and relax a little the conditions imposed in the original MMASS, and allow that agents communicate independently of location (in the original MMASS, agents could only interact including communicate with other agents at neighboring nodes). An agent can interact with objects which are at the same node where it is, performing operations allowed for each object (open a box, kick a ball, pick an object, drop an object, etc.). An agent can emit fields the source node is necessarily the node where the agent is, the starting moment is the moment in which the agent decides to start emitting the field, the initial intensity can be arbitrarily selected by the agent, and of course the field type must be defined in the layer in which the agent is emitting the field. Finally, an agent can move to a neighboring node following the layer topology.

At a time t an agent can sense the fields that are hitting it and receive messages from other agents, trigger the appropriate internal mechanisms to reason about its state, goals, history, etc. and decide a course of actions, which can include messages sent to other agents, interactions with inanimate objects, emission of fields and displacement. As a consequence, at time $t + 1$ the world must be updated: fields are updated, the internal states of agents are updated, messages reach their recipients, the effects of interactions with objects are registered, and displacements occur. If some sort of conflict occurs regarding displacements, e.g. if two agents decide to go to the same node at time t, some sort of *conflict resolution* is triggered. A very simple form of conflict resolution can be as follows: one layer is elected to be the referee (e.g. the one representing geometrical relations), and in case of conflict the agent with shortest label (i.e. the one that is closer to the target node) wins the race, and the other agent(s) stay where it(they) is(are). If there is a draw (e.g. if two agents are equally distant from the target node) then one agent is arbitrarily chosen to win the race.

In very few words, it is a dynamic model of agent interactions together with the structuring of space provided by the layer topology and contextual information provided by the fields emitted in each layer. Some standard tricks can be used to provide agents with useful information, e.g. we can build lamp posts, which are agents that never move and continually emit a special sort of field. Based on lamp posts, the other agents can locate themselves geographically.

Operationally, we have a new FOPL theory built for each time t. Based on this theory, the agents reason and decide what actions they shall be obliged and permitted to take. Permitted actions are followed by their corresponding probabilities. Once these actions are determined, they actually take place, thus changing the environment and updating the status of every agent for time $t + 1$ - formally, this is accomplished by updating the FOPL theory of time t so that it becomes the correct theory for time $t + 1$.

We believe that the Coupled MMASS has great potential for applications related to digital entertainment (games, feature movies), ubiquitous computing, and simulations in general.

In the next section we illustrate through a simple example the Coupled MMASS in action.

4 THE COUPLED MMASS IN ACTION

In this section we briefly illustrate the operations of the Coupled MMASS, through a simple example. The example is a simple computer game, in which the computer plays against itself and we can watch the evolution of a match.

We have a five-by-five board, and two teams of two agents each, depicted as **x** and **o** as presented in Figure 2.

The goal of the **x** team is to reach the rightmost column of the board before any agent from the **o** team reaches the leftmost column of the board, and vice-versa. The movements of each agent, however, are constrained by the following rules:

1. Any agent is *permitted* to emit a field at any moment. An emitted field lasts for one clock tick and reaches the whole board. Given that each agent is *permitted* to emit a field, we must also define the probability that each agent actually does so. In our example, we fix it as a probability of 0.2.
2. If an agent of the opposite team is emitting a field, then an agent is *obliged* to stay at the same position for one clock tick.
3. If no agent from the opposite team is emitting a field and the position immediately ahead of an agent is vacant, then the agent is *permitted* (with probability 0.5) to move to that position.
4. If no agent from the opposite team is emitting a field and the position immediately ahead of an agent is occupied, then the agent is *permitted* (with probability 0.5) to move one position sideways, always to its right, and only if the position at its right is vacant. The right of an **x** agent is one position below it, and the right of an **o** agent is one position above it.

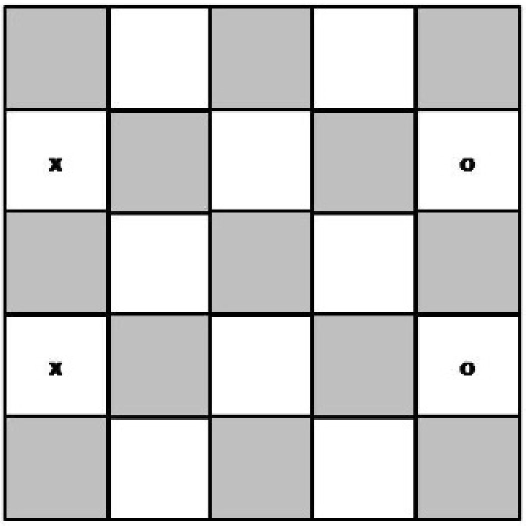


Figure 2. Initial configuration of five-by-five board.

The layer topology for the corresponding Coupled MMASS model for this simple board game coincides with the board itself, i.e. each position is a node, and we have edges connecting each node with its neighbors above, below, to the right and to the left.

We have one physical layer, which corresponds to where the agents are, and one field layer, which is used to represent the existing fields at any clock tick.

We have built a simple (and rather roughly finished) *PROLOG* program to run simulations of this game. The game depicts a radically simple world in which agents behave according to a few general laws and regulations. Despite its simplicity, it can be surprisingly entertaining to watch the non-deterministic simulations.

Our goal with this simple example is to show the potential of the Coupled MMASS framework to build models for the applications referred to in the previous sections.

5 DISCUSSION AND FUTURE WORK

In this article we have introduced a variation of the MMASS framework, coined the Coupled MMASS, in which a simplified version of the original MMASS framework is integrated with a variation of FOPL with non-deterministic proof rules characterizing permissions and obligations of agents to behave according to laws and regulations.

We envisage that the Coupled MMASS framework can be useful to model massively multi-agent systems in which the behavior of agents is governed by complex rules, thus producing complex system behavior that can be best analyzed through computer-based simulations.

Our future work concerning the Coupled MMASS shall focus on three issues:

1. We shall adjust some fine grained details about the framework (e.g. the detailed characterization of proof rules for this framework) and study more carefully some remaining formal issues (e.g. a model theoretic semantics for the framework).
2. We shall work out efficient implementations for the Coupled MMASS, so that it can be effectively employed on practical applications.
3. We shall implement solutions for practical applications, e.g. related to the fields referred to throughout the present article.

ACKNOWLEDGEMENTS

The first author has received financial support from the Brazilian funding agencies CNPq and FAPESP.

REFERENCES

[1] S. Bandini, S. Manzoni, and G. Vizzari, ‘Situated cellular agents: A model to simulate crowding dynamics’, *IEICE Transactions on Simulation and Systems: Special Issue on Cellular Automata*, **E87-D**(3), 669–676, (2004).

[2] Stefania Bandini, Mizar Luca Federici, and Giuseppe Vizzari, ‘Situated cellular agents approach to crowd modeling and simulation’, *Cybernetics and Systems*, **38**(7), 729–753, (2007).

[3] Stefania Bandini, Sara Manzoni, and Giuseppe Vizzari, ‘Multi–agent approach to localization problems: the case of multilayered multi agent situated system’, *Web Intelligence and Agent Systems*, **2**(3), 155–166, (2004).

[4] Michael Batty, *Advanced Spatial Analysis: The CASA Book of GIS*, chapter Agent-based pedestrian modelling, 81–106, Esri Press, 2003.

[5] F.S. Correa da Silva, *On Reasoning With and Reasoning About Uncertainty in Artificial Intelligence*, European Summer Meeting of the Association of Symbolic Logic, Spain, 1996.

[6] F.S. Correa da Silva and W.W. Vasconcelos, *Rule Schemata for Game Artificial Intelligence*, 2006.

[7] M.P. Locatelli and G. Vizzari, ‘Awareness in collaborative ubiquitous environments: The multilayered multi-agent situated system approach’, *TAAS*, **2**(4), (2007).

[8] C. Simone and S. Bandini, ‘Integrating awareness in cooperative applications through the reaction-diffusion metaphor’, *Computer Supported Cooperative Work*, **11**(3-4), 495–530, (2002).

[9] A. Valente, *Legal Knowledge Engineering: A Modeling Approach*, IOS Press, The Netherlands, 1995.

[10] J. Vazquez-Salceda, H. Aldewereld, and F. Dignum, *Implementing Norms in Multi-agent Systems*, 2004.

[11] G. Vizzari, *Dynamic Interaction Spaces and Situated Multi-agent Systems: from a Multi-layered Model to a Distributed Architecture*, PhD Thesis, University of Milano-Bicocca, 2005.

Gwendolen: A BDI Language for Verifiable Agents

Louise A. Dennis [1] [2] and **Berndt Farwer** [3] [4]

Abstract. We describe the Gwendolen BDI (Belief, Desires and Intentions) agent programming language. Gwendolen is implemented in the *Agent Infrastructure Layer* (AIL), a collection of JAVA classes intended for use in model checking agent programs in a variety of languages. The Gwendolen language was developed to test key features of the AIL and its integration with the model checker, JPF, and also to provide a *default semantics* for the AIL classes.

1 Introduction

The AIL (Agent Infrastructure Layer) [Dennis et al., 2008] is a toolkit of JAVA classes designed to support the implementation of BDI (Belief, Desire and Intention) programming languages [Rao and Georgeff, 1995, Wooldridge and Rao, 1999] and the model checking of programs implemented in these languages. Previous approaches to model checking agent programs (e.g., [Bordini et al., 2006]) showed that encoding agent concepts, such as goal and belief, into the state machine of the model checker was a complex and time-consuming task. Similarly it was necessary to adapt the property specification language of a model checker in order to express properties in terms of belief and goals; the natural terminology for reasoning about an agent-based program. Our approach is to encode the relevant concepts from the AIL into the model checker just once and then allow multiple languages to benefit from the encoding by utilising the AIL classes for belief, goal etc., in their implementation. The AIL therefore consists of data structures for representing agents, beliefs, plans, goals and intentions which can be adapted to the operational semantics of individual languages. It has a common agent class intended to function as the base for agent classes in individual languages.

This paper documents the Gwendolen language used during the development of the AIL. It also reports on a simple logic for reasoning about Gwendolen programs using JPF (Java PathFinder) [Visser et al., 2003]. Gwendolen is designed to exhibit many typical features of BDI languages. Programs are presented as a library of plans. Plans are enabled when an agent has certain beliefs and goals and suggest a sequence of deeds to be performed in order to attain a goal. Plans may also be triggered by events, such as changes in belief following perception or the commitment to goals. We term such plans *triggered plans* whereas plans which depend on an agent's internal state alone are *untriggered plans*. Gwendolen agents also distinguish two sorts of goals. *Achievement goals* make statements about beliefs the agent wishes to hold. They remain goals until the agent gains an appropriate belief. *Perform goals* simply state a sequence of deeds to be performed and cease to be a goal as soon as that sequence is complete.

Gwendolen is not intended for use as an actual programming language. The point of the AIL is for many languages to be implemented using it. Of particular interest are the *Jason* [Bordini et al., 2007] implementation of AgentSpeak [Rao, 1996] and 3APL [Dastani et al., 2005], and work is underway to implement these within the AIL. Two existing languages, SAAPL [Winikoff, 2007] and GOAL [de Boer et al., 2007], have already been implemented. Gwendolen does however serve two purposes. Firstly it provides a *default semantics* for the AIL data structures. Any language implemented using the AIL classes will have its own operational semantics and so could, in principle, use the AIL's data structure for perform goals as achievement goals and vice versa. Our intention is that such languages, where possible, should use the rules provided for Gwendolen (the default AIL semantics) only altering these where necessary. Furthermore, where languages change the default semantics it becomes possible to reason about the effects of the changes on model checking. Secondly Gwendolen served a practical purpose as part of the development and debugging of the AIL and has been used in some simple case studies – such as reported in [Dennis and Fisher, 2008].

Gwendolen's agent data structure and the workings of some of the transition rules contain elements which are there purely to assist reasoning about the programs. We will highlight these elements as we encounter them. These suggest adaptations to the semantics of existing agent languages that may be useful to enable reasoning about multi-agent systems.

[1] Dept. of Computer Science, University of Liverpool, Liverpool, UK, L.A.Dennis@liverpool.ac.uk
[2] Work supported by EPSRC grant EP/D052548.
[3] Dept. of Computer Science, Durham University, Durham, UK, berndt.farwer@durham.ac.uk
[4] Work supported by EPSRC grant EP/D054788.

2 Syntax

$$
\begin{array}{rcl}
\mathit{var} & \rightarrow & \text{string beginning with an upper-case letter} \\
\mathit{const} & \rightarrow & \text{string beginning with a lower-case letter} \\
\mathit{term} & \rightarrow & \mathit{var} \\
 & | & \mathit{const} \\
 & | & \mathit{const}(\mathit{term}*) \\
\mathit{literal} & \rightarrow & \mathit{term} \\
 & | & \neg \mathit{term} \\
\mathit{source} & \rightarrow & \mathit{const} \\
 & | & \mathit{var} \\
\mathit{belief} & \rightarrow & \mathit{literal} \\
\mathit{belief}_s & \rightarrow & \mathit{belief}\{\mathit{source}\} \\
\tau_g & \rightarrow & a \\
 & | & p \\
\mathit{goalexp} & \rightarrow & \mathit{literal} \\
\mathit{action} & \rightarrow & \mathit{term} \\
 & | & \mathit{null} \\
\mathit{message} & \rightarrow & (\mathit{const}, \mathit{term})^{\mathit{source}}_{\mathit{const},\mathit{const}}
\end{array}
$$

Messages consist of a pair of a constant and a term (an illocutionary force, and the message content), a source representing the sender of the message, and then two constants for a message id and a thread id.

$$
\begin{array}{rcl}
\mathit{sbelief} & \rightarrow & \mathit{belief}_s \\
 & | & !_{\tau_g} \mathit{goalexp} \\
 & | & \mathit{message}
\end{array}
$$

An *sbelief* (State Belief) is a statement about the agent's internal state. This includes whether the agent has a particular belief, a particular goal, or has sent a particular message.

$$
\begin{array}{rcl}
\mathit{guard} & \rightarrow & \mathit{var} \\
 & | & \mathit{sbelief} \\
 & | & \sim \mathit{sbelief} \\
 & | & \mathit{guard} \wedge \mathit{guard}
\end{array}
$$

A *guard* is a formulae that must be believed by an agent before a plan is applicable. NB. We are using $\neg$ here to represent "believes not" and $\sim$ to represent "doesn't believe".

$$
\begin{array}{rcl}
\mathit{deed} & \rightarrow & \mathit{action} \\
 & | & +\mathit{belief} \\
 & | & -\mathit{belief} \\
 & | & +!_{\tau_g} \mathit{goalexp} \\
 & | & -!_{\tau_g} \mathit{goalexp} \\
 & | & \epsilon
\end{array}
$$

Deeds represent the statements that may appear in the bodies of plan. So an agent may execute an action, add or remove beliefs and add or remove goals. ϵ is a distinguished symbol that is taken to mean "no plan yet".

$$
\begin{array}{rcl}
\mathit{event} & \rightarrow & \mathit{var} \\
 & | & +\mathit{belief} \\
 & | & -\mathit{belief} \\
 & | & +!_{\tau_g} \mathit{goalexp} \\
 & | & \times !_{\tau_g} \mathit{goalexp} \\
 & | & \texttt{start} \\
\mathit{event}_s & \rightarrow & \mathit{event}\{\mathit{source}\}
\end{array}
$$

Events are statements that may trigger a plan. So again this includes changes in belief, commitment to goals, and discovering there is a problem with achieving some goal ($\times !_{\tau_g} \mathit{goalexp}$).

The only point where a programmer may directly refer to the source of a belief or goal is as part of an event or a guard in a plan. This is for matching plans to intentions. A programmer has no control over the assignment of sources to events, beliefs, and so forth.

$$
\begin{array}{rcl}
\mathit{plan} & \rightarrow & (\mathit{event}, \mathit{deed}*) : \mathit{guard} \texttt{ <- } \mathit{deed}* \\
\mathit{plan}_s & \rightarrow & \mathit{plan}\{\mathit{source}\}
\end{array}
$$

Plans consist of an event and deed stack which are matched against the current intention when determining applicability. There is a guard which must succeed for the plan to be applicable and another deed stack, the body of the plan, representing the course of action to be taken.

The initial state of a Gwendolen agent, as specified by a programmer is:

Name:	*const*	**Environment:**	*const*
Beliefs:	*belief* ∗		
Goals:	*goalexp*∗		
Plans:	*plan*∗		

2.1 Notation

In what follows we generally refer to individual beliefs, goals, actions, events, etc. with lower-case letters and sets or stacks of beliefs, goals, actions, etc. with upper-case letters (mostly we presume these are stacks but sometimes we generalise to sets). In general the following letters will be associated with each concept: beliefs (b), goals (g), actions (a), events (e), plans (p), deeds (d) and guards (gu).

3 Functions and Data Structures

3.1 Sets and Stacks

We have generic stack and set data-types with the following constructors, destructors, and operations:

Stack	
$[]$	an empty stack
$x;s$	stack s, with a new top element x
$\texttt{hd}(s)$	top element of stack s
$\texttt{drop}(n, s)$	remove the top n elements from stack s
$\texttt{prefix}(n, s)$	the top n elements of the stack s
$s[n]$	the n-th element of the stack s
$s_1 \texttt{@} s_2$	s_1 appended to the front of s_2
$\#s$	the number of elements in the stack s.
$\texttt{empty}(s)$	true if s is empty
Set	
$x \in S$	x is in the set S
$S_1 \cup S_2$	the union of sets S_1 and S_2
$S_1 \setminus S_2$	the set S_1 less the set S_2
$in_1(S_1 \times S_2)$	S_1

3.2 Intentions

Definition 1 *An* intention *(i) is an abstract data structure which relates a deed stack to a set of triggering events, unifiers, and a source. The idea is that an intention has a source (for whom the intention is being performed) and that any deed on the stack can be traced to its* triggering event. *Individual deeds are also associated with unifiers for its free variables.*

We do not go into the details of the data structure here but note that it supports the following operations.

Intention	
Δ_i	the deed stack of intention i.
$\mathcal{E}_i$	the event stack of intention i.
$\texttt{tr}(n, i)$	returns the event that triggered the placement of the n-th deed on i
$\theta(n, i)$	returns the unifier associated with the n-th deed in the deed stack
$(e, ds, \theta)@_{\texttt{p}}i$,	joins a stack of deeds ds, to an intention's deed stack such that all deeds in ds are associated with the event e and the unifier θ.
$\texttt{drop}_{\texttt{p}}(n, i)$	removes the top n deeds from the stack and also performs appropriate garbage collection on events so that $\mathcal{E}_i$ will only list events still relevant to the truncated deed stack
$\texttt{drop}_{\texttt{e}}(n, i)$	removes the top n events from the intention and also performs appropriate garbage collection on deeds so that Δ_i will only list deeds still relevant to the truncated event list
$[(e, ds, \theta)]\{s\}$	A new intention with deed stack, ds, associated with the event e, the unifier θ and source, s

On top of these can be defined the following.

$\texttt{hd}_{\text{deed}}(i)$	$\texttt{hd}(\Delta_i)$
$\texttt{hd}_{\text{ev}}(i)$	$\texttt{hd}(\mathcal{E}_i)$
$\theta^{\texttt{hd}(i)}$	$\theta(1, i)$
$\texttt{tl}_{\text{int}}(i)$	$\texttt{drop}_{\texttt{p}}(1, i)$
$(e, d, \theta);_p i$	$(e, [d], \theta)@_{\texttt{p}} i$
$i\text{U}_\theta \theta$	$(\texttt{hd}_{\text{ev}}(i), \texttt{hd}_{\text{deed}}(i), \theta \cup \theta^{\texttt{hd}(i)});_p \texttt{tl}_{\text{int}}(i),$
$i[\theta^{\texttt{hd}(i)}/\theta]$	$(\texttt{hd}_{\text{ev}}(i), \texttt{hd}_{\text{deed}}(i), \theta);_p \texttt{tl}_{\text{int}}(i)$
$[(e)]\{s\}$	$\begin{cases} [(e, [\epsilon], \emptyset)]\{s\} & \text{if } e \text{ is a belief update} \\ [(e, e, \emptyset)]\{s\} & \text{otherwise} \end{cases}$
$\texttt{empty}_{\text{int}}(i)$	$\Delta_i = []$

All aspects of an agent's internal state are annotated with sources:

$\texttt{src}(c)$	returns the source of component c

3.3 Agent State

Definition 2 *An* agent state *is a tuple* $\langle ag, \xi, i, I, Pl, A, B, P, In, Out, RC\rangle$ *where ag is a unique identifier for the agent, ξ an environment, i is the current intention, I denotes all extant intentions, Pl are the currently applicable plans (only used in one phase of the cycle), A are actions executed, B denotes the agent's beliefs, P are the agent's plans, In is the agent's inbox, Out is the agent's outbox, and RC is the current stage in the agent's reasoning cycle. All components of the state are labelled with a source.*

Definition 3 *The* initial state *of an agent defined by*

Name:	ag	**Environment:**	ξ
Beliefs:	bs		
Goals:	gs		
Plans:	ps		

is a state

$$\langle ag, \xi, hd(I), tl(I), \emptyset, \emptyset, B, P, \emptyset, \emptyset, \mathbf{A}\rangle$$

where

$$\begin{aligned} I &= map(\lambda g.\ ([(\texttt{start}, +!_{\tau_g} g, \emptyset)]\{\texttt{self}\}), gs) \\ B &= map(\lambda b.\ b\{\texttt{self}\} bs) \\ P &= map(\lambda p.\ p\{\texttt{self}\} ps) \end{aligned}$$

These pair the aspects of the initial state as specified by the programmer with the source `self`.

For notational convenience we will sometimes use the agent identifier, ag, in an agent state $\langle ag, \xi, i, I, appPlans, A, B, P, In, Out, RC\rangle$ to refer to the whole agent and will refer to individual components as ag_B for the belief base B; ag_{Out} for the outbox Out; ag_A for the actions A; and ag_{Is} for the combined set of all intentions $i;I$. The action stack A is one of the data structures included in a Gwendolen agent entirely to assist in reasoning about agents. It represents actions the agent has performed and can be used to verify that, at the least, an agent believes it has done something.

4 Application and Environment

We assume some functions that applications may override

Overridable functions	
Description	Syntax
select intention	$\mathcal{S}_{\text{int}}(I) = \langle i, I' \rangle$
select plan	$\mathcal{S}_{\text{plan}}(P, i) = p$
relevance of sources	$\mathbf{relevant}(s_1, s_2)$

$\mathcal{S}_{\text{int}}$ and $\mathcal{S}_{\text{plan}}$ default to selecting the first intention/plan in the list (I or P, respectively). The function **relevant** defaults to true.

We also assume that the environment in which the agents run supports several functions:

Environment functions	
Description	Syntax
perform action	$\mathbf{do}(a) = \begin{cases} \theta & \text{if action } a \text{ succeeds} \\ \bot & \text{if action } a \text{ fails} \end{cases}$
get new percepts	$\mathit{newpercepts}(ag)$
percepts to remove	$\mathit{oldpercepts}(ag)$
get messages	$\mathit{getmessages}(ag)$

Where **do** performs an action, *newpercepts* returns any new perceptions since the agent last checked, *oldpercepts* returns anything the agent can no longer perceive and *getmessages* returns any new messages for the agent.

5 State Checking

Gwendolen implements a logical consequence relation $\models$ on guards. This relation relates an agent ag with a pair of a guard and a unifier (if the guard is not ground, the unifier returned by $\models$ is a unifier that makes the statement true). This is the default logical consequence relation provided by the AIL and it is a key component in model checking systems using the AIL. In general the model checking process verifies that an agent satisfies a particular property (such as having a belief, or a goal) if the logical consequence relation succeeds.

The semantics of $\models$ is based on an underlying transition system $\rightarrow_{bc}$. This is more elaborate than required at present but we intend to extend $\models$ with Prolog style reasoning.

$$ag \models \mathit{sbelief}, \theta \equiv \langle \mathit{sbelief}, \emptyset, ag \rangle {\rightarrow_{bc}}^* \langle [], \theta, ag \rangle \quad (1)$$
$$ag \models\sim gu \equiv \neg(ag \models gu, \_) \quad (2)$$

$$ag \models gu_1 \wedge gu_2, \theta_1 \cup \theta_2 \equiv ag \models gu_1, \theta_1 \wedge ag \models gu_2\theta_1, \theta_2 \quad (3)$$

NB. we are using the notation $t\theta$ to represent the application of a unifier θ to a term t.

Checking Beliefs:

$$\frac{b' \in ag_B \qquad \text{unify}(b', b_1\theta_1) = \theta}{\langle b_1;bs_1, \theta_1, ag \rangle \rightarrow_{bc} \langle bs_1, \theta \cup \theta_1, ag \rangle} \quad (4)$$

Checking for Goals:

$$\frac{i \in ag_{Is} \qquad !_{\tau_g} g' \in \mathcal{E}_i \qquad \text{unify}(g', g\theta_1) = \theta}{\langle !_{\tau_g} g;bs_1, \theta_1, ag \rangle \rightarrow_{bc} \langle bs_1, \theta \cup \theta_1, ag \rangle} \quad (5)$$

Checking the outbox:

$$\frac{m \in ag_{Out} \qquad \text{unify}(m, m_1\theta_1) = \theta}{\langle m_1;bs_1, \theta_1, ag \rangle \rightarrow_{bc} \langle bs_1, \theta \cup \theta_1, ag \rangle} \quad (6)$$

6 Planning

Gwendolen has two options during the planning phase of a reasoning cycle. Either the current intention continues to be processed, or it needs new planning. This is represented by the function *appPlans*, yielding the currently applicable plans:

$$\mathit{appPlans}(ag, i) = \mathit{continue}(ag, i) \cup \mathit{match\_plans}(ag, i) \quad (7)$$

The applicable plans are a set of tuples, each representing an alteration to be made to the intention i, of the form $\langle \text{trigger}, \text{newplan}, \text{guard}, \text{length}, \text{unifier} \rangle$. To borrow some terminology from 3APL [Hindricks et al., 1999, Dastani et al., 2005], *plan revision plans* specify that a prefix of the current plan should be dropped and replaced by another while *goal planning plans* specify how the current plan should be extended to plan a sub-goal. We have unified this idea with the use of events (which allow an agent to pursue multiple intentions at once). So the above tuple specifies a triggering event for the new deed stack section, the new deed stack (that replaces the old prefix), the guard for the plan (we want to keep a record of this when using untriggered plans), the length of the prefix to be dropped, and a unifier.

6.1 *continue*

continue processes intentions with pre-existing deed stacks, i.e., where there is no need to start off new sub-goals. However such intentions may also be altered by plan revision plans.

$$\mathit{continue}(ag, i) = \{ \langle \mathtt{hd}_{\text{ev}}(i), \mathtt{hd}_{\text{deed}}(i), \top, 1, \theta^{\mathtt{hd}(i)} \rangle | \\ \mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} \neq \epsilon \} \quad (8)$$

6.2 *match_plans*

$$\mathit{match\_plans}(ag, i) = \\ \mathit{match\_plans}_1(ag, i) \cup \mathit{match\_plans}_2(ag, i) \quad (9)$$

The two sets that make up *match_plans* consist of those generated from all the triggered plans in the agent's plan library:

$$\mathit{match\_plans}_1(ag, i) = \{ \langle p_e, p_d, p_{gu}, \#p_p, \theta \rangle | \\ (p_e, p_p) : p_{gu} \texttt{ <- } p_d\{p_s\} \in ag_P \wedge \#p_p > 0 \\ \wedge \text{unify}(\mathtt{tr}(\#p_p, i) : \mathtt{prefix}(\#p_p, \Delta_i)\theta^{\mathtt{hd}(i)}, p_e : p_p) = \theta_e \\ \wedge ag \models p_{gu}\theta_e, \theta_b \wedge \theta = \theta^{\mathtt{hd}(i)} \cup \theta_e \cup \theta_b \} \quad (10)$$

and all the untriggered plans in the plan library.

$$\mathit{match\_plans}_1(ag, i) = \{ \langle p_e, p_d, p_{gu}, \#\Delta_i + 1, \theta \rangle | \\ (p_e, p_p) : p_{gu} \texttt{ <- } p_d\{p_s\} \in ag_P \wedge \#p_p = 0 \wedge \\ \wedge ag \models p_{gu}, \theta_b \wedge \theta = \theta^{\mathtt{hd}(i)} \cup \theta_b \} \quad (11)$$

Although this looks complex, it is the standard BDI planning mechanism for matching triggers and/or guards. There is a fair amount of book-keeping to keep track of unifiers and the added requirement. Because of the variety of different plan types Gwendolen allows to generate a number for the lines to be removed from the current intention.

7 Gwendolen Operational Semantics

Gwendolen has a reasoning cycle based on 6 stages: **A**, **B**, **C**, **D**, **E**, and **F**. In what follows, we omit all parts of the state not affected by a transition for the sake of readability.

Definition 4 *A* multi-agent system *is a tuple of n agents $\mathcal{A}_i (1 \leq i \leq n)$ and an environment ξ.*

$$\frac{\mathcal{A}_i \rightarrow \mathcal{A}'_i}{\{\mathcal{A}_1, ..., \mathcal{A}_i, ..., \mathcal{A}_n, \xi\} \rightarrow \{\mathcal{A}_1, ..., \mathcal{A}'_i, ..., \mathcal{A}_n, \xi\}} \tag{12}$$

7.1 Stage A: Intention Selection

Rule (13) is the standard rule for selecting a new intention. This covers all intentions except empty intentions:

$$\frac{\neg\mathtt{empty}_{\mathrm{int}}(i) \qquad \mathcal{S}_{\mathrm{int}}(I \cup \{i\}) = (i', I')}{\langle ag, i, I, \mathsf{A}\rangle \rightarrow \langle ag, i', I', \mathsf{B}\rangle} \tag{13}$$

Rule (14) tidies away completed intentions – we stay in stage **A** because it is possible the selected intention is also empty.

$$\frac{\mathtt{empty}_{\mathrm{int}}(i) \qquad \mathcal{S}_{\mathrm{int}}(I) = (i', I')}{\langle ag, i, I, \mathsf{A}\rangle \rightarrow \langle ag, i', I', \mathsf{A}\rangle} \tag{14}$$

7.2 Stage B: Generate applicable plans

Rule (15) uses *appPlans* defined in equation (7) to generate a set of plans deemed applicable by Gwendolen.

$$\frac{appPlans(ag, i) \neq \emptyset}{\langle ag, Pl, \mathsf{B}\rangle \rightarrow \langle ag, Pl', \mathsf{C}\rangle} \tag{15}$$
$$\text{where } Pl' = appPlans(ag, i)$$

Rule (16) applies when there are no applicable plans but the triggering event isn't a goal. The new applicable plan is an empty plan.

$$\frac{appPlans(ag, i) = \emptyset \qquad \mathtt{hd}_{\mathrm{ev}}(i) \neq +!_{\tau_g} g}{\langle ag, i, Pl, \mathsf{B}\rangle \rightarrow \langle ag, i, \{\langle \mathtt{hd}_{\mathrm{ev}}(i), [], 1, \emptyset\rangle\}, \mathsf{C}\rangle} \tag{16}$$

Rule (17) applies if there is no applicable plan for some sub-goal. In this case a problem goal event is posted. NB. This does not immediately cause the goal to be dropped. A response to a problem goal has to be handled by a plan. An obvious default plan is $(\times!_{\tau_g} g, \epsilon) : \top$ `<-` $-!_{\tau_g} g$ which will cause problem goals to be dropped (see (24) and (25)) but this mechanism allows for other responses to problem goals.

$$\frac{appPlans(ag, i) = \emptyset \qquad \mathtt{hd}_{\mathrm{ev}}(i) = +!_{\tau_g} g}{\langle ag, i, I, Pl, \mathsf{B}\rangle \rightarrow \langle ag, i, I, \{\langle \times!_{\tau_g} g, [], 0, \theta^{\mathtt{hd}(i)}\rangle\}, \mathsf{C}\rangle} \tag{17}$$

7.3 Stage C: Select a plan

Plan revision rules may wish to alter the prefix of the deeds on a deed stack. For this reason our applicable plan stage has generated a tuple of a triggering event e, a deed stack, ds, a guard, gu, the length of prefix to be dropped, n, and a unifier, θ. In goal planning the prefix length is just 1 (i.e. it will drop the ϵ (no plan yet) marker from the top of the plan and insert the new plan from the rule) but this allows longer prefixes to be dropped in plan revision.

$$\frac{\mathcal{S}_{\mathrm{plan}}(Pl, i) = (\langle e, ds, gu, n, \theta\rangle) \qquad n > 0}{\langle ag, i, Pl, \mathsf{C}\rangle \rightarrow \langle ag, (e, ds, \theta) @_{\mathrm{p}} \mathtt{drop}(n, i)[\theta^{\mathtt{hd}(i)}/\theta], [], \mathsf{D}\rangle} \tag{18}$$

Rule (19) handles untriggered plans. These are plans that are triggered by the agent's state *alone*, not by any particular triggering event or deed stack. We do not want these appended to the current intention. We treat them as a new intention associated with that agent state coming about.

$$\frac{\mathcal{S}_{\mathrm{plan}}(Pl, i) = (\langle e, ds, gu, 0, \theta\rangle)}{\langle ag, i, I, Pl, \mathsf{C}\rangle \rightarrow \langle ag, [(+\mathtt{st}(gu), ds, \theta)]\{\mathtt{self}\}, i;I, [], \mathsf{D}\rangle} \tag{19}$$

7.4 Stage D: Handle top of the Deed Stack

We start with a rule for handling an empty deed stack. This simply proceeds to the next stage:

$$\frac{\Delta_i = []}{\langle ag, i, \mathsf{D}\rangle \rightarrow \langle ag, i, \mathsf{E}\rangle} \tag{20}$$

7.4.1 Achievement Goals

Rule (21) handles situations where a goal has already been achieved. When we achieve a goal the top unifier is transferred to the next goal down in order to preserve instantiations (using the $\mathtt{U}_\theta$ function we defined earlier).

$$\frac{\mathtt{hd}_{\mathrm{deed}}(i)\theta^{\mathtt{hd}(i)} = +!_a g \qquad ag \models g, \theta_g}{\langle ag, i, \mathsf{D}\rangle \rightarrow \langle ag, \mathtt{tl}_{\mathrm{int}}(i)\mathtt{U}_\theta(\theta^{\mathtt{hd}(i)} \cup \theta_g), \mathsf{E}\rangle} \tag{21}$$

Rule (22) sets up a new achieve sub-goal for planning. It does this by making the sub-goal a new triggering event associated with the "no plan yet" symbol. It leaves $+!_a g$ on the deed stack under ϵ. The idea is that ϵ will be replaced by the deed stack to achieve g and then we test that g has indeed been achieved.

$$\frac{\mathtt{hd}_{\mathrm{deed}}(i)\theta^{\mathtt{hd}(i)} = +!_a g \qquad ag \models\sim g}{\langle ag, i, \mathsf{D}\rangle \rightarrow \langle ag, (+!_{\tau_g} g, \epsilon, \theta^{\mathtt{hd}(i)});_p i, \mathsf{E}\rangle} \tag{22}$$

7.4.2 Perform Goals

Rule (23) sets up a new perform sub-goal for planning, as for (22).

$$\frac{\mathtt{hd}_{\mathrm{deed}}(i)\theta^{\mathtt{hd}(i)} = +!_p g}{\begin{array}{c}\langle ag, i, \mathsf{D}\rangle \rightarrow \\ \langle ag, (+!_p g, \epsilon, \theta^{\mathtt{hd}(i)});_p(\mathtt{hd}_{\mathrm{ev}}(i), null, \theta^{\mathtt{hd}(i)});_p \mathtt{tl}_{\mathrm{int}}(i), \mathsf{E}\rangle\end{array}} \tag{23}$$

The *null action* (equivalent to "do nothing"), represented by the placeholder *null*, is required to ensure that we do not lose the record of the event that placed the perform goal on the event stack. This is an example of an operational rule governed by the need to reason about the agent's state. Since we use the event stack to, for instance, reason about the goals of the system we do not want to lose a record of these goals. We remove the perform goal deed from the deed stack, $\mathtt{tl}_{\text{int}}(i)$, but we do not want to lose the the event $\mathtt{hd}_{\text{ev}}(i)$ from the event stack. Since if it only occurs once we lose the record that the agent committed to it (if it was a goal).

7.4.3 Dropping Goals

When dropping a goal we remove all the events on the event stack after we committed to the goal:

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = -!_{\tau_g} g \qquad \text{unify}(+!_{\tau_g} g, \mathcal{E}_i[n]\theta_e(n,i)) = \theta_e \qquad \forall m < n.\ \mathcal{E}_i[m]\theta_e(m,i) \neq +!_{\tau_g} g}{\langle ag, i, \mathbf{D}\rangle \to \langle ag, \mathtt{drop}_{\mathtt{e}}(n,i), \mathbf{E}\rangle} \tag{24}$$

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = -!_{\tau_g} g \qquad \neg\text{unify}(+!_{\tau_g} g, \mathcal{E}_i[n]\theta_e(n,i))}{\langle ag, i, \mathbf{D}\rangle \to \langle ag, \mathtt{tl}_{\text{int}}(i), \mathbf{E}\rangle} \tag{25}$$

7.4.4 Updating Beliefs

Rule (26) adds beliefs. It also starts a new intention triggered by the "new belief" event. This allows for any belief inference that follows from the change.

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = +b}{\langle ag, i, I, B, \mathbf{D}\rangle \to \langle ag, \mathtt{tl}_{\text{int}}(i)\mathtt{U}_\theta\theta^{\mathtt{hd}(i)}, [(+b)]\{\mathtt{src}(i)\};I, B \cup \{b\}, \mathbf{E}\rangle} \tag{26}$$

Rule (27) is for removing beliefs:

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = -b \qquad \exists b' \in B.\text{unify}(b', b) = \theta \qquad \mathbf{relevant}(\mathtt{src}(b'), \mathtt{src}(b))}{\langle ag, i, I, B, \mathbf{D}\rangle \to \langle ag, \mathtt{tl}_{\text{int}}(i)\mathtt{U}_\theta(\theta^{\mathtt{hd}(i)} \cup \theta), [(-b)]\{\mathtt{src}(i)\};I, B \setminus \{b'\}, \mathbf{E}\rangle} \tag{27}$$

7.4.5 Actions

Rule (28) covers generic actions. We use $\uparrow^{ag} m$ for the action of sending a message m to an agent ag. For presentational reasons $a \neq \uparrow$ means $a \neq \uparrow^{ag} (ilf, m)^{ag'}_{m_{\text{id}}, th_{\text{id}}}$

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = a \quad a \neq \uparrow \quad a \neq null \quad \mathbf{do}(a) = \theta_a}{\langle ag, i, A, \mathbf{D}\rangle \to \langle ag, \mathtt{tl}_{\text{int}}(i)\mathtt{U}_\theta(\theta^{\mathtt{hd}(i)} \cup \theta_a), a;A, \mathbf{E}\rangle} \tag{28}$$

Note how we place the action a on the action stack A to record its execution for reasoning purposes.

Sending involves generating a new message id. It is possible the send refers to an old message id (th_{id} for thread id) if it is a reply.

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = \uparrow^{ag} (ilf, \phi)^{ag'}_{,th_{\text{id}}'} \qquad m_{\text{id}} = \mathcal{M}_{\text{id}}(ag, ag') \qquad (th_{\text{id}}' = null \wedge th_{\text{id}} = \mathcal{TH}_{\text{id}}(ag, ag')) \vee th_{\text{id}} = th_{\text{id}}' \qquad \xi.\mathbf{do}(\uparrow^{ag'} (ilf, \phi)^{ag}_{m_{\text{id}}, th_{\text{id}}}) = \theta_a}{\begin{array}{c}\langle ag, \xi, i, I, A, Out, \mathbf{D}\rangle \to \\ \langle ag, \xi, \mathtt{tl}_{\text{int}}(i)\mathtt{U}_\theta(\theta^{\mathtt{hd}(i)} \cup \theta_a), \\ [(\uparrow^{ag'} (ilf, \phi)^{ag}_{m_{\text{id}}, th_{\text{id}}}, \epsilon, \theta^{\mathtt{hd}(i)} \cup \theta_a)]\{\mathtt{self}\};I, \\ \uparrow^{ag'} (ilf, \phi)^{ag}_{m_{\text{id}}, th_{\text{id}}};A, Out \cup \{\uparrow^{ag'} (ilf, \phi)^{ag}_{m_{\text{id}}, th_{\text{id}}}\}, \mathbf{E}\rangle\end{array}} \tag{29}$$

where $\mathcal{M}_{\text{id}}$ and $\mathcal{TH}_{\text{id}}$ are functions that generate fresh id numbers for messages and threads respectively.

There is a rule for null actions:

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = null,}{\langle ag, i, \mathbf{D}\rangle \to \langle ag, \mathtt{tl}_{\text{int}}(i)\mathtt{U}_\theta\theta^{\mathtt{hd}(i)}, \mathbf{E}\rangle} \tag{30}$$

Lastly we have two rules for unsuccessful actions:

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = a, \qquad \mathtt{hd}_{\text{ev}}(i) = +!_{\tau_g} g \qquad \neg\xi.\mathbf{do}(a)}{\langle ag, \xi, i, I, \mathbf{D}\rangle \to \langle ag, \xi, (\times!_{\tau_g} g, \epsilon, \theta^{\mathtt{hd}(i)});_p i, I, \mathbf{E}\rangle} \tag{31}$$

$$\frac{\mathtt{hd}_{\text{deed}}(i)\theta^{\mathtt{hd}(i)} = a, \qquad \mathtt{hd}_{\text{ev}}(i) \neq +!_{\tau_g} g \qquad \neg\xi.\mathbf{do}(a)}{\langle ag, \xi, i, I, \mathbf{D}\rangle \to \langle ag, \xi, i, I, \mathbf{E}\rangle} \tag{32}$$

7.5 Stage $\mathbf{E}$: Perception

$$\frac{}{\langle ag, \xi, i, I, [], \mathbf{E}\rangle \to \langle ag, \xi, i, I' \mathbin{@} I'' \mathbin{@} I, In, \mathbf{F}\rangle} \tag{33}$$

where $In = \xi.getmessages(ag)$,
$I' = \{[((+b, +b, \emptyset))]\{\mathtt{percept}\} \mid b \in \xi.newpercepts(ag)\}$,
and $I'' = \{[((-b, -b, \emptyset))]\{\mathtt{percept}\} \mid b \in \xi.oldpercepts(ag)\}$

Perception does not directly act on the belief base, instead it sets up intentions to alter the belief base – this allows for planning based on, for instance, the reliability of the information.

7.6 Stage $\mathbf{F}$: Message Handling

$$\frac{}{\langle ag, i, In, \mathbf{F}\rangle \to \langle ag, I' \mathbin{@} I, [], \mathbf{A}\rangle} \tag{34}$$

where $I' = \{[(+ \downarrow^{ag} (ilf, \phi)^{ag'}_{m_{\text{id}}, th_{\text{id}}})]\{ag'\} | (ilf, \phi)^{ag'}_{m_{\text{id}}, th_{\text{id}}} \in In\}$

We use $\downarrow^{ag} m$ to represent the fact that agent, ag has received message, m. Again the receipt of messages doesn't directly act on the belief base, but causes new intentions to be set up noting the receipt of the message.

8 Communication Semantics

Gwendolen has no fixed semantics for communication instead both receiving and sending messages are treated as belief update events and start new intentions. This allows a semantics of communication for any particular program to be established using plans. For instance a semantics for **perform** messages can be established with the following plan:

$$(+ \downarrow^{A_1} (\mathbf{perform}, Goal)^{A_2}_{M_{\text{id}}, Th_{\text{id}}}, \epsilon) : \top \texttt{ <- } +!_p Goal$$

This specifies that the content of a message with a **perform** illocutionary force should be established as a perform goal.

We also intend to extend the AIL and Gwendolen with grouping mechanisms such as described in [Dennis et al., 2007] with the intention that these can be used to program a range of multi-agent coordination models such as organisations and roles [Ferber et al., 2003], joint intentions [Cohen and Levesque, 1991] and institutions [Esteva et al., 2001].

9 Model Checking

The AIL includes a set of interfaces which allow a dedicated set of classes, referred to as the *MCAPL (Model Checking Agent Programming Languages) interface* to model check programs. The MCAPL interface allows a user to specify simple properties in a temporal and modal logic and then check that these properties hold using the JPF (Java PathFinder) model checker. At present we can verify properties of simple programs written in Gwendolen using properties expressed in the following *MCAPL property specification language*:

$$\begin{array}{rl} a ::= & \text{constant} \\ f ::= & \text{ground first order formula} \\ \phi ::= & \mathbf{B}(ag, f) \mid \mathbf{G}(ag, f) \mid \mathbf{A}(ag, f) \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \\ & \mid \phi\, \mathbf{U}\, \phi \mid \phi\, \mathbf{R}\, \phi \end{array} \tag{35}$$

Some of these formulas have a semantics determined by the implementation of the MCAPL interface. The AIL implements the formulas $\mathbf{B}(ag, f)$ and $\mathbf{G}(ag, f)$ as *sbelief*s. Consider a program P for a multi-agent system and let MAS be the state of the multi-agent system at one point in the run of the program. Consider an agent, $ag \in MAS$, at this point in the program execution, $MAS \models_{MC} \mathbf{B}(ag, f)$ iff $ag \models f, \theta$ (for some θ). Similarly AIL interprets $\mathbf{G}(ag, f)$ to mean that $MAS \models_{MC} \mathbf{G}(ag, f)$ iff $ag \models !_a f, \theta$ (for some θ). We use the syntax f : *belief* and f : *action* here to show how the logical consequence relation distinguishes between the formulae based on their type. By contrast reasoning about executed actions inspects the action stack so $MAS \models_{MC} \mathbf{A}(ag, f)$ iff $f \in ag_A$.

The other formulas in the MCAPL property specification language have an LTL (Linear Temporal Logic)-based semantics defined by the MCAPL interface and including the operators until ($\mathbf{U}$) and release ($\mathbf{R}$).

$$MAS \models_{MC} \phi_1 \wedge \phi_2 \text{ iff } MAS \models_{MC} \phi_1 \text{ and } MAS \models_{MC} \phi_2$$
$$MAS \models_{MC} \phi_1 \vee \phi_2 \text{ iff } MAS \models_{MC} \phi_1 \text{ or } MAS \models_{MC} \phi_2$$
$$MAS \models_{MC} \neg\phi \text{ iff } MAS \not\models_{MC} \phi.$$

The temporal formulas hold true of runs of the programs in the JPF model checker. A run consists of a sequence of program states $MAS_i, 0 \leq i \leq n$ where MAS_0 is the initial state of the program and MAS_n is the final state in the run. Let P be a multi-agent program $P \models_{MC} \phi_1 U \phi_2$ iff in all runs of the program there exists a program state MAS_j such that $MAS_i \models_{MC} \phi_1$ for all $0 \leq i < j$ and $MAS_j \models_{MC} \phi_2$. Similarly $P \models_{MC} \phi_1 \mathbf{R} \phi_2$ iff either $MAS_i \models_{MC} \phi_1$ for all, i or exists MAS_j such that $MAS_i \models_{MC} \phi_1$ for all, $0 \leq i \leq j$ and $MAS_j \models_{MC} \phi_1 \wedge \phi_2$. The more commonly recognised temporal operators $\Diamond$ (eventually) and $\Box$ (always) are special cases of $\mathbf{U}$ and $\mathbf{R}$.

10 Current Status

We have written a number of simple programs based on "Blocks World"-type examples where agents form goals to pick up blocks, or to get other agents to pick up blocks. Two such programs are shown below:

10.1 Program 1

This is a single agent program. The agent believes a block to be available and that its hands are empty. The has a single achievement goal – to pick something up. It has a single plan, triggered by the desire to pick something up and guarded by something being available to pick up. It then adds a belief that it has picked up the object and removes the belief that its hands are empty.

$$\begin{array}{rl} \textbf{Name}: & ag1 \\ \textbf{Beliefs}: & empty \\ & available(block) \\ \textbf{Goals}: & !_a pickup(X) \\ \textbf{Plans}: & (+!_a pickup(Y), \epsilon) : empty \wedge available(Y) \\ & \qquad\qquad \texttt{<-}\ +pickup(Y); \\ & \qquad\qquad\qquad -empty \end{array}$$

We can verify properties of this such as $\Diamond(\mathbf{B}(ag1, pickup(block)))$. It is also possible to establish the falsity of expressions such as $\Box(\neg(\mathbf{B}(ag1, pickup(block)) \wedge \mathbf{B}(ag1, empty)))$. This property is violated because the agent first adds the belief it has picked up the block *before* it removes the belief that its hands are empty.

10.2 Program 2

This program consists of two agents, a master and a slave. The master has a goal, to pick something up. To achieve this it sends a message to the slave with the illocutionary force, **achieve**, telling the slave to pick something up.

The slave has two plans. The first of these performs a pick up action if it wants to pick something up, while the second establishes the semantics for **achieve**, namely that the slave establishes an achievement goal when asked to achieve something.

$$\begin{array}{rl} \textbf{Name}: & master \\ \textbf{Beliefs}: & check \\ \textbf{Goals}: & !_a pickup \\ \textbf{Plans}: & (+!_a pickup, \epsilon) : \top \\ & \quad \texttt{<-}\ \uparrow^{slave} (\textbf{achieve}, pickup)^{master}_{M_{\text{id}}, Th_{\text{id}}}; \\ & \qquad wait; \\ & \qquad +!_a check \\ \textbf{Name}: & slave \\ \textbf{Plans}: & (+!_a pickup, \epsilon) : \top\ \texttt{<-}\ pickup \\ & (+\downarrow^{slave} (\textbf{achieve}, Goal)^{master}_{M_{\text{id}}, Th_{\text{id}}}, \epsilon) : \top\ \texttt{<-}\ +!_a Goal \end{array}$$

We can verify some properties of this program, such as $\Diamond(\mathbf{B}(master, check))$. But, because JPF does not assume a fair scheduler, it is impossible to verify, for instance, $\Diamond(\mathbf{B}(master, pickup))$ because there is a run where the slave agent never gets an opportunity to do anything. We are currently working to overcome these limitations and to customise JPF to both improve its efficiency in an agent setting and provide a flexible set of appropriate configurations for checking agent programs under a range of assumptions.

11 Conclusions

We have presented here a prototype language, Gwendolen, used in developing the AIL, which is a collection of classes intended to assist in model checking agent programs written in a variety of languages. We have discussed how the MCAPL interface allows us to inspect a Gwendolen agent in order to deduce properties of that agent for use in model checking.

Gwendolen provides a default semantics for the AIL classes. This allows an implementation of an existing language using the AIL to (potentially) prove that it has preserved the semantics of the AIL data structures sufficiently to generate sound results from the model checking process.

The construction of the Gwendolen language has also revealed places where language semantics need to be specialised if we intend to reason about agents written in that language.

12 Further Work

In future we intend to add basic constructs for forming groups to Gwendolen based on a framework outlined in [Dennis et al., 2007] and at the same time to extend the MCAPL property specification language with appropriate primitives for discussing groups of agents.

We also hope to customise JPF for model-checking with agent languages. For instance, if we assume that the program states of interest in the model-checking process are those that occur after every rule application (for instance) rather than every Java state executed then we can increase the complexity of programs that can be verified within a reasonable time.

Lastly we intend to implement a number of existing BDI languages in the AIL (two implementations, SAAPL [Winikoff, 2007] and GOAL [de Boer et al., 2007] are already complete with others, in particular AgentSpeak [Bordini et al., 2007] underway) and investigate correctness issues.

REFERENCES

[Bordini et al., 2006] Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2006). Verifying Multi-Agent Programs by Model Checking. *J. Autonomous Agents and Multi-Agent Systems*, 12(2):239–256.

[Bordini et al., 2007] Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-agent Systems in AgentSpeak Using Jason.* Wiley.

[Cohen and Levesque, 1991] Cohen, P. R. and Levesque, H. J. (1991). Teamwork. Technical Report 504, SRI International, California, USA.

[Dastani et al., 2005] Dastani, M., van Riemsdijk, M. B., and Meyer, J.-J. C. (2005). Programming Multi-Agent Systems in 3APL. In Bordini, R. H., Dastani, M., Dix, J., and Seghrouchni, A. E. F., editors, *Multi-Agent Programming: Languages, Platforms and Applications*, pages 39–67. Springer.

[de Boer et al., 2007] de Boer, F. S., Hindriks, K. V., van der Hoek, W., and Meyer, J.-J. C. (2007). A Verification Framework for Agent Programming with Declarative Goals. *J. Applied Logic*, 5(2):277–302.

[Dennis et al., 2008] Dennis, L. A., Farwer, B., Bordini, R. H., and Fisher, M. (2008). A flexible framework for verifying agent programs (short paper). In Padgham, Parkes, Muller, and Parsons, editors, *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008).* ACM. To Appear.

[Dennis and Fisher, 2008] Dennis, L. A. and Fisher, M. (2008). Programming Verifiable Heterogeneous Agent Systems. In *Proc. Eighth International Workshop on Programming Multiagent Systems (ProMAS).* To Appear.

[Dennis et al., 2007] Dennis, L. A., Fisher, M., and Hepple, A. (2007). Foundations of flexible multi-agent programming. In *Eighth Workshop on Computational Logic in Multi-Agent Systems (CLIMA-VIII).*

[Esteva et al., 2001] Esteva, M., Rodríguez-Aguilar, J. A., Sierra, C., Garcia, P., and Arcos, J. L. (2001). *Agent Mediated Electronic Commerce. The European AgentLink Perspective*, chapter On the Formal Specificaion of Electronic Institutions, pages 126–147. Number 1991 in LNAI. Springer.

[Ferber et al., 2003] Ferber, J., Gutknecht, O., and Michel, F. (2003). From Agents to Organizations: An Organizational View of Multi-agent Systems. In *Proc. 4th International Workshop on Agent-Oriented Software Engineering (AOSE)*, volume 2935 of *LNCS*, pages 214–230. Springer.

[Hindricks et al., 1999] Hindricks, K. V., de Boer, F. S., van der Hoek, W., and Meyer, J.-J. C. (1999). Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401.

[Rao, 1996] Rao, A. (1996). AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *Proc. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*, volume 1038 of *LNCS*, pages 42–55. Springer.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. (1995). BDI Agents: from theory to practice. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, San Francisco, CA.

[Visser et al., 2003] Visser, W., Havelund, K., Brat, G. P., Park, S., and Lerda, F. (2003). Model Checking Programs. *Automated Software Engineering*, 10(2):203–232.

[Winikoff, 2007] Winikoff, M. (2007). Implementing Commitment-Based Interactions. In *Proc. 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1–8, New York, NY, USA. ACM.

[Wooldridge and Rao, 1999] Wooldridge, M. and Rao, A., editors (1999). *Foundations of Rational Agency.* Applied Logic Series. Kluwer Academic Publishers.

Theory and Practice of Social Reasoning Experiences with the iCat

Frank Dignum, Bas Steunebrink, Nieske Vergunst, Christian Mol, John-Jules Meyer[1]

Abstract. In this paper we discuss how a social companion robot can be programmed using agent technology. An important challenge is to combine the physical and conceptual elements of social relationships.

1 INTRODUCTION

The term "social reasoning" can be used in many different ways and in many different settings. One can argue that if one is planning a holiday with the family, social reasoning is needed to find a destination that is, in some sense, socially optimal. For this kind of reasoning one could use game theory or something related. Our setting, however, is of a different nature. We aim to program a sociable robot on the iCat, developed by Philips. The iCat is a static robot (no legs or wheels) that can only move his head and create facial expressions. Some pictures of the expressions can be seen in figure 1.

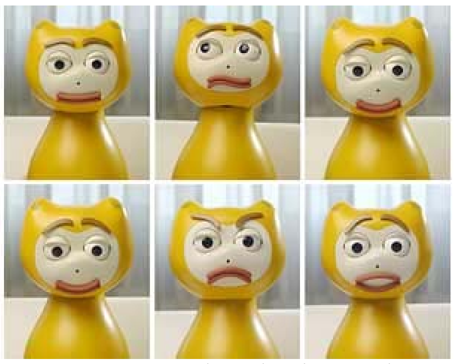


Figure 1. iCat showing emotions

The iCat can see through a camera in its nose and hear through microphones in its paws. It also has a proximity sensor in his left paw and touch sensors in its paws and ears. Besides the facial expressions the iCat can also talk or send information through a network link to other devices (in this way it can e.g. switch on a tv or a lamp).

The iCat platform comes with OPPR which allows you to create animations in a simple way using timelines and/or LUA scripts. OPPR also takes care of lip sync for speech output and of the blending of different outputs such as when the iCat has to simultaneously smile and talk.

In our project we are using the iCat as a kitchen companion that can assist persons in the kitchen with cooking, making shopping lists, etc. In this setting we need social reasoning in order to find e.g. a recipe that suits the whole family, but we also need social reasoning over the physical, real-time interaction. For instance, the iCat can give a complete recipe in one time, but probably the person needs to have instructions spread out during the cooking period (at the time they are needed for the next step in the preparation). And even on a lower level the iCat should check whether the person pays attention or is getting bored (by means of e.g. gaze tracking) with a conversation in order to switch to another mode.

We assume that for a high level of social behaviour the robot should be conscious of goals, desires, values, preferences, etc. of itself and the person and also how these relate (e.g. common goals, common beliefs, contradictory goals and beliefs). This led us to the use of the agent paradigm as a means to model and implement the reasoning on the iCat. Agents also are defined in terms of goals, intentions, plans and beliefs and thus seem a very good fit.

However, the traditional agent architecture can best be depicted as in figure 2. This follows the sense-reason-act loop. In such a loop the agent senses the environment, updates its beliefs and based on its current goals and intentions checks which action it should perform next.

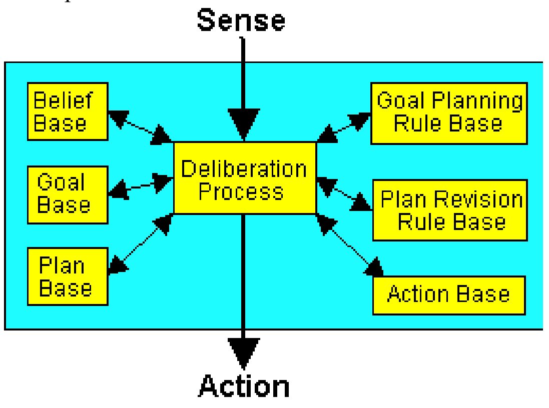


Figure 2. architecture of BDI agents

In this architecture the reasoning part is autonomous and cannot be interrupted. It is a well-known fact that this feature can lead to big problems in real-time environments where quick reactions to changing situations are necessary.

The most well-known social robots Max and Kismet [1,3] have thus been implemented mainly using a reactive architecture in which the robot directly responds based on the inputs it gets from the environment (although in Max [3] also goals are used and in Kismet [1] "urges" play a role).

Since real-time aspects are very important for social behaviour and thus for social robotics we are trying to combine

[1] Dept. of Information and Computing Science, Utrecht University, The Netherlands. Email: {dignum,bass,nieske,Christian,jj}@cs.uu.nl.

the traditional BDI architecture with the more reactive approaches that already exist. In the next section we sketch a short overview of the architecture and will explain shortly which role the social reasoning takes here and how this is modelled in this architecture.

2 ARCHITECTURE

In figure 3 we show the architecture that is used to model the social robot as we are implementing it on the iCat.

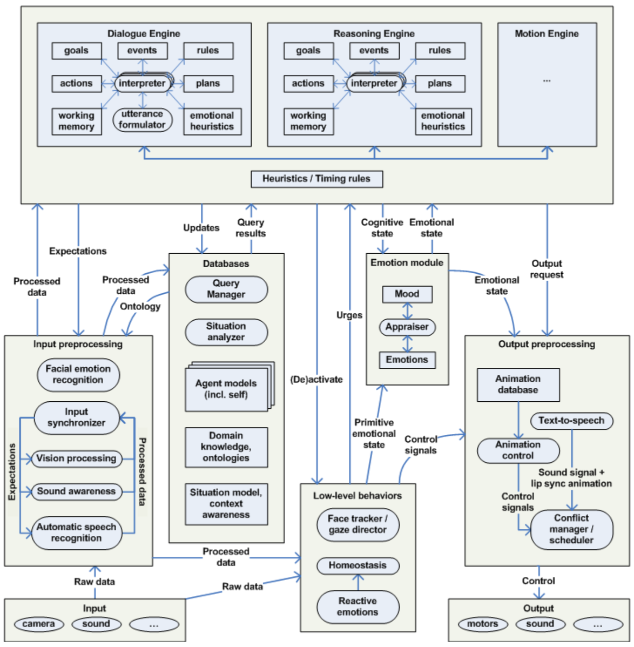


Figure 3. architecture of a social robot

Although we will not discuss the architecture in detail (we have another paper in preparation for that [4]) the figure illustrates nicely that the architecture of a social robot is a lot more complex than that of a traditional agent. This complexity stems from the combination of the following two points:

1. Social behaviour encompasses many different elements that should be combined.
2. Real-time aspects are important for social behaviour.

As an example of point one, a social robot has to interpret and give social cues. In order to interpret a social cue the robot should be aware of the emotions of the user, have a user profile and history, have domain knowledge and be aware how its position is in both the physical and social relation with the user. If one takes only the first point into account, one could still use a traditional agent architecture. This architecture could have a complex belief base encompassing models of the user(s), domain knowledge, history, etc. However, the second point indicates that there is no unlimited time to deliberate on its actions. For many groups in robotics, the real time aspect prompted them to start of with a reactive system that could respond in time to an event. Of course this was one of the reasons for Brooks to propose a layered architecture in which each layer builds on a (reactive) layer below [2].

The way we try to solve the problem of combining a very rich deliberation process with real-time aspects is by splitting up the different elements of the social reasoning in separate components that can run in parallel but interact through a few main deliberation loops. On the highest level we distinguish three deliberations. One for maintaining the dialogue with a user, one for general goal achievement and one for motor behaviour. The last one is just drawn for completeness as it is not needed for the iCat which only can move (some parts of) its head.

These deliberation cycles are connected with lower level processes on both the input and output side. Besides these two traditional parts we also formed this into a kind of hybrid Brooks architecture and have a low-level reactive module that takes care of primary reactions and urges such as hunger, sleep, etc.

The main challenge in using this architecture is not so much the deliberation processes on the highest level, but determining which data to feed into these high level deliberation processes and what to handle in the other modules. E.g. it is clear that one wants some geometric reasoning to take place on the vision input. This can be used to reason that a ball is moving. However, does one just indicate that the ball is moving or also the change in position or speed? Depending on what the deliberation loop is doing different types of data might be important at different intervals. If the robot is contemplating to intercept the ball, it probably wants to know as much as possible about the position and speed changes of the ball. But if it is actually talking with the user about dinner, the ball is only a distraction and no data is needed except that the ball is moving.

Our thesis is that social reasoning depends for a large part in determining how to zoom in and zoom out of the different processes that comprise the social situation and combine the results of these processes in a coherent way. This can be (partly) achieved by using things like “focus of attention” and maintaining a “shared situation awareness”. Both terms are however not very precisely definable and need more exploration to be truly useful.

7 CONCLUSIONS

We have sketched an architecture for social robots and tried to indicate the position of social reasoning from the perspective of this architecture. The work is an ongoing effort and we need more experience with an implementation to be able to sustain our claim that this architecture helps in modelling and implementing social robots. An important aspect for us is the question how to actually program a robot like this. Which parts of the architecture should be domain independent, which programming language should be used for the different modules and how should they be connected.

REFERENCES

[1] C. Breazeal. Designing Sociable Robots. MIT press (2002).
[2] R. Brooks. Intelligence without representation. *Artificial Intelligence* 47, pp. 139-159, (1991)
[3] S. Kopp, L. Gesellensetter, N. Krämer, I. Wachsmuth . A conversational agent as museum guide -- design and evaluation of a real-world application. Panayiotopoulos et al. (eds.): *Intelligent Virtual Agents*, LNAI 3661, pp. 329-343, Springer-Verlag, (2005).
[4] B. Steunebrink et.al. A generic Architecture for a Companion Robot. (submitted to MARS 2008)

How can machines reason?

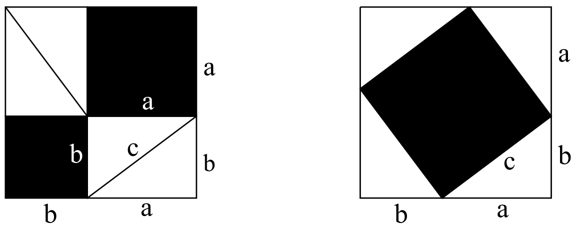
Mateja Jamnik[1]

1 Can machines reason like humans in mathematics?

Some of the deepest and greatest insights in reasoning were made using mathematics. It is not surprising therefore that emulating such powerful reasoning on machines – and particularly the way humans use diagrams to "see" an explanation for mathematical theorems – is one of the important aims of artificial intelligence.

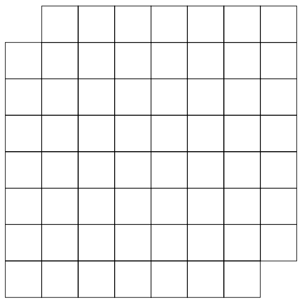
2 Diagrams for reasoning

Drawing pictures and using diagrams to represent a concept is perhaps one of the oldest vehicles of human communication. In mathematics, the use of diagrams to prove theorems has a rich history: as just one example, Pythagoras' Theorem has yielded several diagrammatic proofs (one is given in the figure below) in the 2500 years following his contribution to mathematics, including that of Leonardo Da Vinci's 2000 years later. These diagrammatic proofs are so clear, elegant and intuitive that with little help, even a child can understand them [9].



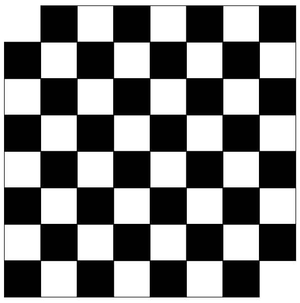
$$a^2 + b^2 = c^2$$

The concept of the "mutilated" checkerboard is another useful demonstration of how intuitive human reasoning can be used to solve problems. If we remove two diagonally opposite corner squares (like in the picture of the mutilated checkerboard in the figure below):



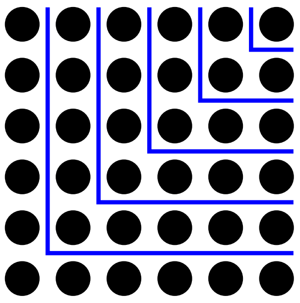
[1] University of Cambridge Computer Laboratory, UK, Web: www.cl.cam.ac.uk/~mj201

can the board still be covered with dominoes (that is, rectangles made out of two squares)? The elegant solution is to colour the checkerboard with alternative black and white squares, like the chessboard (in the figure below),



and do the same with the dominoes so that a domino is made of one black and one white square. The solution then immediately becomes clear: there are more black squares than white squares, and so the mutilated checkerboard cannot be covered with dominoes. This problem is very easy for people to understand, but no system has yet been implemented that can solve it in such an elegant and intuitive way.

Here is another example of an "informal" diagrammatic proof: that of a theorem about the sum of odd natural numbers. Unlike in automated theorem proving, where this problem is tackled symbolically with formal logical tools, the diagrammatic operations of our proof enable us to "see" the solution: a square can be cut into so-called "L" shaped pieces (as in the figure below), and each "L" represents a successive odd number, since both sides of a square of size n are joined ($2n$), but the joining vertex was counted twice (hence $2n - 1$).



$$n^2 = 1 + 3 + 5 + \cdots + (2n - 1)$$

These examples above demonstrate how people use "informal" techniques, like diagrams, when solving problems. As these reasoning techniques can be incredibly powerful, wouldn't it be exciting if a system could learn such diagrammatic operations automatically? So far, few automated systems have attempted to benefit from their power by imitating them [5, 1, 2, 4]. One explanation for this might be that we don't yet have a deep understanding of informal techniques and how humans use them in problem solving. To advance the state of the art of automated reasoning systems, some of these informal human reasoning techniques might have to be integrated with the proven successful formal techniques, such as different types of logic.

3 From intuition to automation

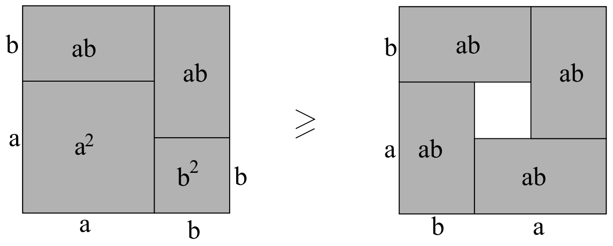
There are two approaches to the difficult problem of automating reasoning. The first is cognitive, which aims to devise and experiment with models of human cognition. The second is to approach the problem computationally – attempting to build computational systems that model part of human reasoning.

In my research at the University of Cambridge, I have taken steps along the computational approach. While at the University of Edinburgh, I built a system known as Diamond that uses diagrammatic reasoning to prove mathematical concepts [6, 8]. Diamond can prove theorems such as the one above about the odd natural numbers using geometric operations, just like in the example. With Diamond I showed that diagrammatic reasoning about mathematical theorems can be automated. Diamond uses simple and clear concrete rather than ambiguous general diagrams (in the example above, the concrete example is for $n = 6$). The "inference steps" of the solution in Diamond are intuitive geometric operations rather than complex logical rules. The universal statement of the theorem is showed using concrete cases by means of particular logical rules. The solution in Diamond is represented as a recursive program, called a schematic proof, which given a particular diagram, returns the proof for that diagram [7, 3]. To check that this solution is correct, Diamond carries out in the background a verification test in an abstract theory of diagrams.

Diamond currently only tackles theorems which can be expressed as diagrams. However, there are theorems, like the mutilated checkerboard, that might require a combination of symbolic and diagrammatic reasoning steps to prove them, so-called heterogeneous proofs. The figure below demonstrates a heterogeneous proof. The theorem states an inequality: $\frac{a+b}{2} \geq \sqrt{ab}$ where $a, b \geq 0$. The first few symbolic steps of the proof are:

$$\begin{array}{rcl}
\frac{a+b}{2} & \geq & \sqrt{ab} \\
 & \downarrow & \text{square both sides of } \geq \\
\frac{(a+b)^2}{2^2} & \geq & ab \\
 & \downarrow & \times 4 \text{ on both sides of } \geq \\
(a+b)^2 & \geq & 4ab \\
 & \downarrow & \\
a^2 + 2ab + b^2 & \geq & 4ab
\end{array}$$

The second part of the proof, which is presented in the figure below, shows diagrammatically the inequality $a^2 + 2ab + b^2 \geq 4ab$.



In Cambridge, I am now investigating how a system could automatically reason about such proofs. This requires combining diagrammatic reasoning in Diamond with symbolic problem solving in an existing state-of-the-art automated theorem prover. The way forward is to give such a heterogeneous reasoning framework access to intelligent search facilities in the hope that the system will not only find new and more intuitive solutions to known problems, but perhaps also find new and interesting problems.

Automated diagrammatic reasoning could be the key not only to making computer reasoning systems more powerful, but also to providing the necessary tools to study and explore the nature of human reasoning. We might then have a means to investigate the amazing ability of the human brain to solve problems.

ACKNOWLEDGEMENTS

The research reported in this paper was supported by EPSRC Advanced Research Fellowship GR/R76783.

REFERENCES

[1] D. Barker-Plummer and S.C. Bailin, 'Proofs and pictures: Proving the diamond lemma with the GROVER theorem proving system', in *Working Notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, ed., N.H. Narayanan, Cambridge, MA, (1992). American Association for Artificial Intelligence, AAAI Press.

[2] J. Barwise and J. Etchemendy, *Hyperproof*, CSLI Press, Stanford, CA, 1994.

[3] A. Bundy, M. Jamnik, and A. Fugard, 'What is a proof?', *Philosophical Transactions of Royal Society A: The Nature of Mathematical Proof*, **363**(1835), 2377–2391, (2005).

[4] *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, eds., B. Chandrasekaran, J. Glasgow, and N.H. Narayanan, AAAI Press/MIT Press, Cambridge, MA, 1995.

[5] H. Gelernter, 'Realization of a geometry theorem-proving machine', in *Computers and Thought*, eds., E. Feigenbaum and J. Feldman, 134–152, McGraw Hill, New York, (1963).

[6] M. Jamnik, *Mathematical Reasoning with Diagrams: From Intuition to Automation*, CSLI Press, Stanford, CA, 2001.

[7] M. Jamnik and A. Bundy, 'Psychological validity of schematic proofs', in *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, eds., D. Hutter and W. Stephan, number 2605 in Lecture Notes in Artificial Intelligence, 321–341, Springer Verlag, (2005).

[8] M. Jamnik, A. Bundy, and I. Green, 'On automating diagrammatic proofs of arithmetic arguments', *Journal of Logic, Language and Information*, **8**(3), 297–321, (1999).

[9] R.B. Nelsen, *Proofs without Words: Exercises in Visual Thinking*, Mathematical Association of America, Washington, DC, 1993.

Open Problems in Simulation and Story Analysis

Ethan Kennerly[1]

Abstract. The game designer is charged with an aesthetic imperative. The design is partitioned into four channels: simulation, user interface, story, and look and feel. Of these, the simulation and story channels are considered. In a game, the aesthetics of a simulation and of a dramatic story are suspected to be deeply coupled in conflict, choice, and change. Using the mechanics-dynamics-aesthetics framework, the simulation and story channels are isolated to survey *practical problems* encountered when designing a game, including dramatic elements of the player's choice, conflict, and change.

1 INTRODUCTION

x	x	o
o	o	
x	o	x

Table 1. A game board of Tic-tac-toe on the turn of X

In the game board of Tic-tac-toe, how many moves does the user have? The player of X has only one move, but the user has at least one more option. She can quit playing. In fact, on every turn the user always has the option to quit playing. While game theory usually has no need to consider this alternative, quitting play is the ultimate problem that game designers face. In non-interactive entertainment, such as a dramatic movie, the audience, too, has the option to physically leave or mentally excurse on a daydream. Along the lines of decision theory, the user is trying to satisfy an aesthetic preference for entertainment (over boredom). The user not only has a choice of moves to make in the game, but may also choose, when none of the moves promise to be entertaining, to quit playing and seek entertainment elsewhere. If the system requirements and the user requirements have been met, then the game has failed to entertain. Out of all the options available to the user, the game has ceased to be the most engaging. Perhaps the game session engendered frustration or, as in the Tic-tac-toe example, boredom.

Hereafter, I shall *italicize phrases that indicate problems designers face*, for which formal techniques would be a boon. As a designer in need, I cannot claim competence for their solution. Instead this survey is intended to promote collaboration between designers and theorists.

The Tic-tac-toe user's boredom is derived from having solved the problem posed by the simulation dynamics. In the Tic-tac-toe example above, it does not matter whether the pieces of the user interface are ergonomic or whether the endgame cinematic has an epic look and feel. To facilitate this distinction when analyzing the design of a game, over the past few years I have found it helpful to partition the game design into four channels: simulation, user interface, story, and look and feel.

	gameplay	**style**
inferred	simulation	story
perceived	user interface	look and feel

Table 2. Four channels of game design

Here is a brief definition and example of each channel, using the ancient Chinese boardgame, Go.

- ***simulation***: The abstract rules governing play. In Go's simulation, a new stone's *position* may not be in an occupied *cell.*
- ***user interface***: The input controls and the output representation to the human players, abstracted from any fine art. In Go's user interface, there is an orthogonal *grid*, and several black and white *markers* intended to be placed on the intersections.
- ***story***: The premise, characters, and plot that the user is internally narrating. In Go's story, a group is *alive* or *dead* and may be *attacked* or *defended.*
- ***look and feel***: The sensorial style and artistry, such as stylistic qualities of visual, aural, and tactile senses. In the look and feel of a fine Go set, the board is thick and made of golden *kaya* wood, the black stones are *slate* and the white are *clamshell* with faint wavy lines on one side.

In a manner somewhat similar to Jesse James Garrett's dual representation of interface and information [1], all four channels contribute to the user's experience. Analyzing a game into these channels isolates phenomena under discussion and invites a multidisciplinary analysis of games, which leverages the decades to centuries of expertise in discrete mathematics, ergonomics, dramatic writing, and the sensorial arts. Design discovers a consonance among the channels.

In this article, only the channels of simulation and story will be discussed, because of the wealth of prior theoretical work. By a dual analysis of the simulation and story channels, we will see that channel consonance is a corequisite to entertainment.

Here is an example of simulation and story consonance. At DigiPen, a team of students created their senior project, Narbacular Drop. In the premise of its story, Princess No-knees escapes a devious dungeon. In the salient feature of its simulation, the avatar maps space on arbitrarily placed portals to advance through puzzles. The simulation and story compatibility is slightly dissonant; the user's belief is stretched to accept a princess that shoots teleporters. In the follow-up, Portal, the salient feature of the simulation is the same, but Portal's premise of its story is: A human test subject completes a robotic lab's training. The story's premise of a science fiction laboratory fits better with the simulation mechanics of space-warp puzzles. So

[1] Interactive Media, School of Cinematic Arts, University of Southern California, United States `kennerly@finegamedesign.com`

the simulation and story channels are interdependent, and in the case of Portal, they are consonant.

Designers, who must combine simulation and story, would benefit from *a model that unifies the aesthetics of a game's simulation and story* [2, 3].

2 AESTHETIC REQUIREMENTS

In 2005, Gregor Benedetti had evoked a fearful look and feel in HeroCard Nightmare. But the simulation and story channels of the game were not evoking a consonant sense of tension and fear. So the project was cancelled. Hating to see the evocative art and components go to waste, I proposed a redesign of the simulation and story channels, while preserving the user interface and look and feel channels. Benedetti's horrific look and feel set an aesthetic requirement of fear and suspense. I faced the practical problem: What simulation and story dynamics heighten tension and induce an entertaining form of fear?

Because the general problem of *correlating a user's emotional response to a game's simulation mechanics* is germane to game design, Marc LeBlanc proposed partitioning the design into: mechanics, dynamics, and aesthetics [4]. Following each definition is an example in the simulation channel and story channel of Go.

- ***mechanics***: the rules and assets of the game, as an artifact of development. In Go's simulation mechanics, if a *contiguous set* of stones has no *empty* neighbors, then that set of stones is *eliminated.* In Go's story mechanics, if a *group* loses all its *liberties* the group is *killed.*
- ***dynamics***: the game session created from the execution of the mechanics and its interaction with users. In Go's simulation dynamics, suppose that during play a *maximally connected* set of stones only has one empty neighbor, and so might be eliminated during the next move. In Go's story dynamics, her *dumpling* is about to be *eaten.*
- ***aesthetics***: the emotional experience by the users. In Go's simulation aesthetics, suppose a player is *worried* about diminishing their *points* by the *count* of stones eliminated. In Go's story aesthetics, suppose a player is *indignant* at having *her* dumpling eaten.

The overview suggests *a functional relationship: mechanics influence dynamics, which influence aesthetics*. The users have unique beliefs and preferences. For that reason, designers empathize with the user, and aim to understand the user's aesthetic preferences, even when the user does not understand them [5].

Aesthetics has long been a domain of interest among dramatists. Egri restates a classic model of aesthetic requirements as a drama's subjective experience, which consists of a rising action, climax, and conclusion [6]. In a dramatic story, Egri posits that the aesthetics correlate to the dynamics of the story. Let us look at the story dynamics and consider their possible confluences with simulation dynamics.

3 DYNAMICS OF SIMULATION AND STORY

Egri explains the dynamics of drama by fundamentals often referred to as character, conflict, choice, and change. Due to a broad scope of media and attention to the psychology of motivation, I will adapt Egri's concepts to games.

Personality and objective makes a character more interesting. In God of War, the bloodlust of Kratos elicits an emotional response, albeit immature and for a juvenile aesthetic profile.

The characters in a drama are pitted in conflict. In Half-Life, Gordon's lab is invaded by aliens. In Ico, Ico and Yorda are trapped in a castle and pursued by shadows.

The choices made by the characters drive their story. Gordon Freeman explores an alien-infested laboratory. Ico escapes a demonic castle.

The result of the choice that the character makes changes the values and beliefs of the character, in narratives and games. The psychology of Macbeth changes from loyal to murderous. The isolation of Ico from Yorda reveals his, and the target user's, need for companionship.

To elevate tension, writers sometimes employ dynamics of risk to craft the character's options into a dilemma. The greatest such dilemma in a story is the crisis. In the season one finale of the television drama, Heroes, Peter and Nathan Petrelli face the crisis of destroying New York or committing seeming suicide.

Let us apply these dramatic terms to the simulation.

3.1 Representing a simulation session as a dramatic story

Sid Meier's most popular game, Civilization, emphasizes the aesthetics of the simulation channel. In the management of an empire, the user is often considering the dynamics of the simulation as a "series of interesting choices."

Part of the interest in a choice is due to the "risk and return" [7]. The user risks his or her resources or chances of winning in exchange for the prospect of greater resources. Here is a modified example from Super Mario Bros. At the beginning of World 5-1, the user may take a low-risk option of destroying a Koopa Troopa (turtle) by fireball, or perform the higher risk maneuver of jumping onto and then kicking the turtle shell to hit several enemies in a combo. If the user's timing or trajectory is mistaken, then the avatar might lose a life to the Koopa Troopa. To compensate for the higher risk, expert execution yields an extra life. So with sufficient skill, the risk of loss is offset by the gain. For another example in Reiner Knizia's two-player card game Lost Cities, before pursuing an expedition a player may play an investment card. The investment card multiplies the points of the expedition, which may turn out to be negative or positive. *Correlating the simulation dynamics of risk and return to the aesthetic experience of play*, such as excitement, would refine the design of simulation mechanics.

Part of the interest in a choice is due to the drama. Marc LeBlanc applies the aesthetics of a dramatic arc to the card game Magic: The Gathering. During the opening game, players have few resources, so the intensity of conflict is relatively low. The players draw resources for attacking and defending through mechanics that conform to a producer-consumer pattern [8]. Thereby the uncertainty and inevitability increase until it is obvious which player will win. Then tension diminishes [4]. Likewise, in Go, Poker, StarCraft, and Civilization, resources are produced over time, through producer-consumer mechanics. For the target users of these kinds of games, the dynamics of risk emerging from such mechanics often induce an aesthetic experience that adheres to a dramatic arc [4].

3.2 Modeling a dramatic story as a simulation session

The requirement of rational agents with perfect knowledge is incompatible with storytelling. From Shakespeare to soap opera, drama is about characters with imperfect intellects [6]. Addressing this, Lowe and Pacuit formally model mistaken beliefs and preferences in a dramatic story [9]. In their analysis of an example story, they represent the characters as players in a formal game, who make choices that lead to conflict and whose conclusion changes the beliefs of at least one of the characters. This model elucidates a fundamental connection between the channels of story and simulation. If their techniques could be adapted to *model the aesthetic consequences of changes in belief,* then it might help integrate the design of a game's simulation and story.

Although a game's drama cannot be reduced to a narrative [2], and the emotions of videogames differ from those of cinema [10], the simulation and story share similar characters, conflict, choice, and change. Egri analyzes dramatic conflict as opponents attacking and counterattacking. Lowe and Pacuit analyze a story as tactical scheming. Modeling the player of a game in Egri's dramatic terms and Lowe and Pacuit's terms of beliefs, it is reasonable to conjecture: *The player, facing a conflict, makes the main character's choices and through the conclusion of the most difficult choice, the crisis, experiences a change in values or beliefs*. To develop this hypothesis, let us discuss some special cases.

4 DRAMA: CORRELATING DYNAMICS TO AESTHETICS

Conflict is crafted by balancing player skill and scenario challenge [11, 12]. By design the difficulty and risk of problems with a similar solution, which Inoue and Ushijima call a gimmick, is ramped. In Super Mario Bros, the gimmick of a cliff is iterated in four stages so the user may: remember, practice, apply, and master that gimmick. In their example, a cliff is leapt from four times: first, with ground below, second without, third with a performance evaluation at the flagpole, and fourth with a moving platform [13]. Designers sequence the difficulty and risk to teach an essential skill of play while developing a dramatic arc, derived from challenging the player. This design task would benefit from *a theoretical relationship between challenge construction, skill acquisition, and the aesthetics of drama*.

One special case of the dramatic conflict has been formalized in deterministic turn-based games of perfect information. In combinatorial game theory, the heat of a game state correlates the simulation's dynamics to the urgency of making a choice [14]. In Go, this is often a situation when one or both players can claim initiative, such as during a ko fight. It would be interesting to investigate *whether the combinatorial game theoretic heat of a simulation state correlates to the aesthetic experience of its users*.

In the dynamics of the simulation channel, there have been several mathematical analyses, some of which could be suspected to have correlations to the aesthetic quality of the game's users. For instance, Chess has too many draws at the championship level [15]. An extremal case of this trend is Tic-tac-toe, or any other game that a player has solved. The result of every session is predictable, and therefore changes no beliefs.

A videogame can induce a sudden change in belief that results in surprise [16]. Emiliano Lorini and Cristiano Castelfranchi have formally modeled surprise as an epistemic change [17]. While their model is beyond my comprehension, if it is compatible with the simulation dynamics of a game that modifies player belief, this epistemic model of surprise would offer a special case solution to the problem of correlating simulation dynamics to aesthetics.

For some classes of game simulations, some parts of the dynamics have been formalized. To give HeroCard Nightmare a simulation dynamics-induced sense of suspense and fear, I inverted the salient simulation mechanics of Clue. In Nightmare's simulation channel, you begin with a deal of cards that when discovered, eliminates you from the game. In the story channel, you are shown a photograph of where you will die (your dealt scene card), and who will kill you (your dealt killer card). Together, these *simulation and story mechanics induce dynamics of bluffing correlating to aesthetics of fear*. Probability of discovery increases during play, increasing fear, while disseminating disinformation may mislead players, incentivizing bluffing. Moving someone else's killer pawn away might deceive other players into misidentifying your actual killer. I have an intuitively derived and empirically tested correlation of Nightmare's simulation dynamics to the aesthetics of its user experience. What I did not have was a theory of knowledge games. Hans P. van Ditmarsch modeled the salient mechanics of Clue (or Cluedo) as a knowledge game, in which players are dealt cards, and winning consists of knowing the deal of the cards [18]. In doing so, there is a formal model of the dynamics of play. If the theory of *knowledge games could be extended to model correlations to a user's aesthetic experience*, then the aesthetics of a game like Clue could be discussed analytically.

In both the dynamics of a simulation channel and a story channel of some dramatic games, choice reveals character and advances the conflict toward a conclusion. In a model like Lowe and Pacuit's, a choice reveals information of a player's preferences, and therefore adjusts the beliefs of observers. The actions of a player in Clue, Poker, or Go reveals the aspects of the strategy and beliefs of the active player. This applies to many genres of games.

It is hard to imagine a satisfying simulation channel in which the enthusiastic player does not have some belief about the simulation dynamics or about another player altered through the course of play. Seeing a newly dealt card in Seven-card Stud Poker or Lost Cities changes belief on the strength of one's hand. This change in belief may alter tactics. Discovering a weakness in one's wall of stones in Go may stimulate the player to protect the weakness before it is exploited. Such cases of play often induce excitement among users. It would be a boon to designers if the dynamics of belief change, such as the work of Ditsmartch, Lorini and Castelfranchi, could be adapted to model a user's emotional state.

5 CONCLUSIONS

A player (in the simulation channel) and a character (in the story channel) changes or reveals preferences. The player-character can be jointly analyzed as a simulation-story. For instance, Ico's, and the user's, preferences for companionship, are revealed when Yorda is lost. Lowe and Pacuit have taken steps to model the revelation of preferences in a simulation-story.

This article has not formalized techniques, which are beyond the author's competence. Instead, I have *italicized practical problems* that designers face without theoretical tools. Centuries ago, Cardano, Pascal, and Fermat formalized the analysis of simulation mechanics in games of chance, clarifying the design of countless dice and card game mechanisms to follow. Even if the simulation and story problems are not solvable, partial solutions would advance the art of crafting games that satisfy our aesthetic requirements.

ACKNOWLEDGEMENTS

Thanks to Benedikt Lowe whose preliminary discussion inspired this paper and informed the recent work on the formal analysis of games and stories. Thanks to the anonymous referees for their detailed comments, which have eliminated numerous errors and digressions. The remaining are my fault. If you know of discoveries along the lines investigated herein, I would appreciate and acknowledge your bringing them to attention.

REFERENCES

[1] J. J. Garrett, *The Elements of User Experience: User-Centered Design for the Web*, New Riders Press, Indianapolis, Indiana, United States, 32-34 (2002).
[2] H. Jenkins. Game Design as Narrative Architecture, *The Game Design Reader: A Rules of Play Anthology*, MIT Press, Cambridge, United States, 670-689 (2006).
[3] J. Schell, Understanding entertainment: story and gameplay are one, In: *Computers in Entertainment*, ACM, New York, **3**(1), 6-6 (2005).
[4] M. LeBlanc, Tools for Creating Dramatic Game Dynamics, *The Game Design Reader: A Rules of Play Anthology*, MIT Press, Cambridge, United States, 438-459 (2006).
[5] O. Sotamaa, Perceptions of Player in Game Design Literature, In: *Situated Play, Proceedings of DiGRA 2007 Conference*, A. Baba (Ed.), University of Tokyo, Tokyo, 456-465 (2007).
[6] L. Egri, *The Art of Dramatic Writing: Its Basis in the Creative Interpretation of Human Motives*, Simon and Schuster, New York, 2nd edn. (1960).
[7] M. Sakurai, Game Design: Risk and Return, Game Developers Conference 2004, San Jose, California, United States (2004).
[8] S. Bjork, J. Holopainen, *Patterns In Game Design*, Charles River Media, Hingham, Massachusetts, United States, 111-112 (2005).
[9] B. Lowe, E. Pacuit, An abstract approach to reasoning about games with mistaken and changing beliefs. Submitted to *Australasian Journal of Logic,* Australasian Association for Logic (Under review).
[10] B. Bates, H. Barwood, L. Sheldon, M. Pondsmith, P. Pizer, Story Summit, Game Developers Conference 2003, San Jose, California, United States (2003).
[11] T. Fullerton, C. Swain, and S. Hoffman, *Game Design Workshop: Designing, Prototyping, and Playtesting Games*, CMP Books, San Francisco, California, United States, 81 (2004).
[12] P. Lemay, Developing a pattern language for flow experiences in video games, *Situated Play, Proceedings of DiGRA 2007 Conference*, A. Baba (Ed.), University of Tokyo, Tokyo, 449-455 (2007).
[13] A. Inoue, S. Ushijima, Around Sigeru Miyamoto: Enactment of "Creator" on Computer Games (sic), In: *Situated Play, Proceedings of DiGRA 2007 Conference*, A. Baba (Ed.), University of Tokyo, Tokyo, 498-502 (2007).
[14] E. Berlekamp, J. Conway, R. Guy, *Winning Ways for your Mathematical Plays*, vol. 1, A K Peters Limited, Natick, Massachusetts, United States, 2nd edn., 151 (2001).
[15] J. D. Beasley, *The Mathematics of Games*, Oxford University Press, Oxford, 145 (1989).
[16] J. Frome, Eight Ways Videogames Generate Emotion, In: *Situated Play, Proceedings of DiGRA 2007 Conference*, A. Baba (Ed.), University of Tokyo, Tokyo, 831-835 (2007).
[17] E. Lorini, C. Castelfranchi, The cognitive structure of Surprise: looking for basic principles. In: *Topoi: An International Review of Philosophy*, Springer, Netherlands, **26**(1), 133-149 (2007).
[18] H. P. van Ditmarsch, Knowledge Games, Dissertation Series, Institute for Logic, Language & Computation, University van Amsterdam, Amsterdam (2000).

A Perception Oriented MAS Model with Hybrid Commonsense Spatial Reasoning

Stefania Bandini and **Alessandro Mosca** and **Matteo Palmonari** and **Giuseppe Vizzari** [1]

Abstract. This paper presents an approach to the integration of a logic based approach to the extension of behavioural specifications in a reactive model. In fact, in general, reactive models provide a robust support to the modelling and implementation of agent's environment, they often lack in supporting the definition of complex autonomous agent behaviours. In particular the paper will describe how the Hybrid Commonsense Spatial Logic approach to spatial reasoning can improve the definition of behavioural rules for agents in the Multilayered Multi-Agent Situated System (MMASS) model. The two models are briefly introduced, then a description of the integration of the two approaches is given, with examples related to agents spatial perception.

1 Introduction

Reactive agent models are very often based on indirect interaction models [13], which support a poor agent internal architecture and very simple behavioural specification mechanisms (e.g. stimulus-action rules). These models generally provide a relevant role of the environment in agent coordination and interaction, but in general also in their decisions on their actions. From the Influence-Reaction model, to pheromone [20] and computational field [17] based approaches, reactive agent models have provided clear definitions of notions like locality and perception. Using the head-body metaphor described in [23], these models provide a robust support to the modelling, design and implementation of agents' bodies and environment.

Nonetheless their simple agent architecture might also represent a limit in the definition of complex and autonomous behaviours. While very complex overall system behaviour may be obtained as an emergent effect of local actions and interactions of very simple agents [19], it might be necessary or even only convenient to be able to exploit higher level expressions, typical of deliberative approaches. The latter, in fact, were often focused on the definition of models, techniques, languages supporting the definition of agents that reason explicitly on facts related their knowledge of the environment, in addition to conceptualizations of their goals, plans or intentions (see, e.g., Belief Desire and Intention [22] models).

However, very often these models overlook the notion of perception, and in general the interaction among the agent and the environment in which it is placed. While there are some approaches that provide the definition of a sort of environmental ontology and mechanisms supporting the transition from environment conditions to concepts on which agents can reason [10], this paper proposes a different approach.

The Multilayered Multi-Agent Situated Systems (MMASS) [2] will be introduced as a multi-agent model providing an explicit spatial structure representation and a set of interaction mechanisms that are strongly related to agents' context. Exploiting this structure and these mechanisms it is possible to define systems of agents which can obtain complex form of coordination by means of a simple behavioural specification.

However, although it is quite simple to model a situation in which an agent is able to move in a spatial structure according to signals that are spread in it by elements of the environment modelled as static agents, it is not so simple to encapsulate into MMASS action specification the deliberative process that governs the autonomous choice of an agent, i.e. to be sensitive to a specific type of signal, and thus its choice to move towards a certain destination. Basically agents follow their perceptions and they have a rather reactive flavor, whereas actual modelling experiences, i.e. in the simulations domain [4], showed that their decision making process could take advantages from a more structured knowledge of the environment. This leads to investigate the possible integration among a perception-based multi-agent model and commonsense spatial reasoning from a knowledge representation perspective.

The aim of this paper is therefore to present the integration of MMASS with a model-based logical approach to spatial reasoning in order to provide some of the MMASS agents with a more rational behavior, giving them the opportunity to exploit an explicit model of the environment in their action selection process. The MMASS model is introduced in the next section, with respect to its basic elements and concepts and to formal definitions. Section 3 presents a formal model supporting commonsense spatial reasoning; since this Commonsense Spatial Model (CSM) is defined as a relational structure, it is shown how it can be viewed as the semantics specification for a hybrid modal language. The integration of the MMASS model with the CSM is discussed in section 4, investigating the motivations, the mapping between the two model and some reasoning issues. Concluding remarks end the paper.

2 Multilayered Multi-Agent Situated System Model

2.1 An Overview of MMASS Model

MMASS provides an explicit model of the environment, that is a graph of sites on which agents are situated. Every site may host at most one agent (according to a non-interpenetration principle), and every agent is situated in a single site at a given time (non–ubiquity principle). Agents inherit the spatial relationships defined for the site

[1] Department of Informatics, Systems and Communication, University of Milano-Bicocca, Italy, email:{bandini, alessandro.mosca, matteo.palmonari, vizzari}@disco.unimib.it

it is occupying; in other words an agent positioned in site p is considered adjacent to agents placed in sites adjacent to p.

The adjacency relation among agents is a necessary condition for the applicability of *reaction*, the first kind of interaction mechanism defined by the MMASS model. This interaction mechanism involves two or more agents that are placed in adjacent sites and allows them to synchronously change their state, after they have performed an agreement. The second interaction mechanism defined by the MMASS model provides the possibility for agents to emit *fields*, that are signals able to diffuse through the environment that can be perceived by other agents according to specific rules. Fields may convey more complex kind of information than just their intensity value, moreover for every field type a *diffusion function* can be specified in order to define how related signals decay (or are amplified) during their diffusion in the environment, from the source of emission to destination sites. Other functions specify how fields of the same kind can be *composed* (for instance in order to obtain the intensity of a given field type at a given site) or *compared*. From a semantic point of view fields themselves are neutral even if they can have related information in addition to their intensity; they are only signals, with an indication on how they diffuse in the environment, how they can be compared and composed. Different agent types may be able to perceive them or not and, in the first case, they may have completely different reaction, according to their behavioural specifications. An agent may perceive a field with a non–null intensity active in the site it is situated on according to two parameters characterizing its type and related to the specific field type. The first one is the *sensitivity threshold*, indicating the minimum field intensity that an agent of that type is able to perceive. The second is the *receptiveness coefficient* and it represents an amplification factor modulating (amplifying or attenuating) field value before the comparison with the sensitivity threshold. Thanks to these parameters it is possible to model dynamism in the perceptive capabilities of agents of a give type, since these parameters are related to agent state. In this way, for instance, the same agent that was unable to perceive a specific field value could become more sensitive (increase its own receptiveness coefficient) as a consequence of a change in its state. This allows to model physical aspects of perception, but also conceptual ones such as agent interests.

Reaction and field emission are two of the possible actions available for the specification of agent behaviour, related to the specification of how agents may interact. Other actions are related to the possibility to move (*transport* operation) and change the state upon the perception of a specific field (*trigger* operation). These primitives are part of a language for the specification of MMASS agents behaviour [2]. An important part of the language also provides the possibility to dynamically modify the structure of agent environment, in order to generate new sites and edges (or destroy existing ones) and create (or destroy) agents of a specific type, with a given initial state. *Agent type* is in fact a specification of agent state, perceptive capabilities and behaviour.

The model has been successfully applied to several simulation contexts in which the concepts of space and environment are key factors for the problem solving activity and cannot be neglected (e.g. crowd modelling [4], localization problems [3]). In the ubiquitous computing context, instead, active entities of a pervasive system (e.g. users having a computational device, sensors and other sources of information) can be modelled as agents which interact by means of an infrastructure which is somehow mapped to a spatial structure (see, e.g., [5]). This structure should reflect the dynamics of the actual environment, and thus it should dynamically determine the possible interactions among agents according to spatial relationships (e.g. distance). The MMASS model allows the representation of these elements (i.e. active entities and their environment) in a unified framework. In the following the model will be briefly introduced and formally described.

2.2 MMASS: Formal Definitions

A *Multilayered Multi–Agent Situated System (MMASS)* is defined as a constellation of interacting *Multi-Agent Sistuated System* (MASS) that represent different layers of the global system: $\Big\langle MASS_1, \ldots, MASS_n \Big\rangle$. A single MASS is defined by the triple $\Big\langle Space, F, A \Big\rangle$ where *Space* models the environment where the set A of agents is situated, acts autonomously and interacts through the propagation of the set F of fields and through reaction operations.

The structure of a layer is defined as a not oriented graph of sites. Every *site* $p \in P$ (where P is the set of sites of the layer) can contain at most one agent and is defined by the 3–tuple $\Big\langle a_p, F_p, P_p \Big\rangle$ where:
• $a_p \in A \cup \{\bot\}$ is the agent situated in p ($a_p = \bot$ when no agent is situated in p, i.e. p is empty);
• $F_p \subset F$ is the set of fields active in p ($F_p = \emptyset$ when no field is active in p);
• $P_p \subset P$ is the set of sites adjacent to p.

In order to allow the interaction between different MMASS layers (i.e. intra-MASS interaction) the model introduces the notion of *interface*. The latter specifies that a gateway among two layers is present with reference to a specific field type. An interface is defined as a 3–tuple $\Big\langle p_i, p_j, F_\tau \Big\rangle$ where $p_i \in P_i, p_j \in P_j$, with P_i and P_j sets of sites related to different layers (i.e. $i \neq j$). With reference to the diffusion of field of type F_τ the indicated sites are considered adjacent and placed on the same spatial layer. In other words fields of type F_τ reaching p_i will be diffused in its adjacent sites (P_p) and also in p_j.

A MMASS agent is defined by the 3–tuple $< s, p, \tau >$ where τ is the *agent type*, $s \in \Sigma_\tau$ denotes the *agent state* and can assume one of the values specified by its type (see below for Σ_τ definition), and $p \in P$ is the site of the $Space$ where the agent is situated. As previously stated, agent *type* is a specification of agent state, perceptive capabilities and behaviour. In fact an agent type τ is defined by the 3–tuple $\Big\langle \Sigma_\tau, Perception_\tau, Action_\tau \Big\rangle$. Σ_τ defines the set of states that agents of type τ can assume. $Perception_\tau : \Sigma_\tau \rightarrow [\mathbf{N} \times W_{f_1}] \ldots [\mathbf{N} \times W_{f_{|F|}}]$ is a function associating to each agent state a vector of pairs representing the *receptiveness coefficient* and *sensitivity thresholds* for that kind of field. $Action_\tau$ represents the behavioural specification for agents of type τ. Agent behaviour can be specified using a language that defines the following primitives:
• $emit(s, f, p)$: the $emit$ primitive allows an agent to *start the diffusion of field* f on p, that is the site it is placed on;
• $react(s, a_{p_1}, a_{p_2}, \ldots, a_{p_n}, s')$: this kind of primitive allows the specification a *coordinated change of state* among adjacent agents. In order to preserve agents' autonomy, a compatible primitive must be included in the behavioural specification of all the involved agents; moreover when this coordination process takes place, every involved agents may dynamically decide to effectively agree to perform this operation;
• $transport(p, q)$: the $transport$ primitive allows to *define agent movement* from site p to site q (that must be adjacent and vacant);
• $trigger(s, s')$: this primitive specifies that an agent must *change its state* when it senses a particular condition in its local context (i.e.

its own site and the adjacent ones); this operation has the same effect of a reaction, but does not require a coordination with other agents.

For every primitive included in the behavioural specification of an agent type specific preconditions must be specified; moreover specific parameters must also be given (e.g. the specific field to be emitted in an emit primitive, or the conditions to identify the destination site in a transport) to precisely define the effect of the action, which was previously briefly described in general terms.

Each MMASS agent is thus provided with a set of sensors that allows its interaction with the environment and other agents. At the same time, agents can constitute the source of given fields acting within a MMASS space (e.g. noise emitted by a talking agent). Formally, a field type t is defined by $\left\langle W_t, \mathit{Diffusion}_t, \mathit{Compare}_t, \mathit{Compose}_t \right\rangle$ where W_t denotes the set of values that fields of type t can assume; $\mathit{Diffusion}_t : P \times W_f \times P \rightarrow (W_t)^+$ is the diffusion function of the field computing the value of a field on a given space site taking into account in which site (P is the set of sites that constitutes the MMASS space) and with which value it has been generated. $\mathit{Compose}_t : (W_t)^+ \rightarrow W_t$ expresses how fields of the same type have to be combined (for instance, in order to obtain the unique value of field type t at a site), and $\mathit{Compare}_t : W_t \times W_t \rightarrow \{\mathit{True}, \mathit{False}\}$ is the function that compares values of the same field type. This function is used in order to verify whether an agent can perceive a field value by comparing it with the sensitivity threshold after it has been modulated by the receptiveness coefficient.

It must be noted that this basic set of primitives can be suitably exploited to generate significant patterns of behaviour, for instance supporting the modeling of pedestrians, as discussed in [8]. However, this formal model is not the best support to represent and manage in a modular and flexible way the different agent attitudes and goals, such as those driving the movement of a human agent in a large scale scenario. In the case of crowd modeling, we use a finite state automata to represent different attitudes towards movement (e.g. move out of a room into a corridor, move towards the stairs), with transitions among these states (e.g. passed from the office door, arrived at the stairs). This kind of automata can be viewed as a form of representation of knowledge about the environment and agent's goals. This way of defining agents' behaviours, however, is not simple and leads to specifications that are hardly reusable.

Even in simulation scenarios, there are situations in which the assumption that (i) agents own some form of knowledge on the environment in which they are situated and (ii) they can exploit it to decide on the actions to be carried out is a realistic one. Moreover, to supply the modeler with proper abstractions and mechanisms to specify some aspects of agents' behaviours that are not easily captured by simple states and reactive rules would simplify the modeling phase and the specification of agents' behaviours. In the following we will describe a model and an approach to provide additional abstractions supporting a simpler and more effective way of defining agents' behaviours.

3 Commonsense Spatial Model

3.1 Basic Concepts: Places and Conceptual Spatial Relations

The literature about space modeling, allowing the adoption of computational frameworks to develop reasoning capabilities, is wide and distributed in several areas of Artificial Intelligence such as Automated Vision, Robotics, Knowledge Representation. Within a rough classification, two main classes of approaches can be distinguished: a first one tends to justify commonsense spatial inference with mathematical models such as Euclidean geometry, trigonometry, differential equations systems and so on [12]; in the second one different topological approaches can be considered, ranging from point set and algebraic topology, with the choice of different kinds of primitive entities and relationships (e.g. RCC calculus [21], modal logics [1]), to topological route maps (see [15, 16], and [14]).

Within the second conceptual framework, *commonsense spatial reasoning* can be supported by defining a formal model of the environment that exploits the basic notions of place and conceptual spatial relation. This approach was fruitfully proposed to enable context-awareness in pervasive computing environments [7], where spatial disposition of information sources distributed in the environment (e.g. close circuit cameras, smart home or complex industrial plant sensor networks) can be mapped into a set of relations among interesting places (i.e. a topology) and high-level reasoning beyond low-level sensors' capabilities can be carried out by reasoning about properties holding at different places.

What about the relationships among this approach to spatial reasoning and MAS? With respect to pervasive computing systems this approach is justified by the consideration that there is little interest in applying reasoning to obtain a deeper knowledge of the spatial environment for what concerns its morphological aspects, because those aspects are partly known by design and partly computable by means of technological tools. But since our aim is to enable agents with some spatial reasoning capability, and not to build spatial models, e.g. from visual information, the argument for pervasive computing holds as well here. In fact, the MAS model we start from, i.e. MMASS, already provides a model of the environment and the fact that this model is a graph bounds to reason precisely over background graph-like structures. Therefore, if we want to integrate such a perception based model with logic-based reasoning capabilities, we should start from the basic spatial structure available from the model: there exist a given spatial structure but this structure needs to be formally specified from a logical point of view (trivial) and (more interestingly) represented and enriched in order to enable agents' reasoning capabilities.

In the following subsection we will present the notion of Commonsense Spatial Model (CSM) as a relational structure, analyzing the formal properties of the commonsense spatial relations involved. It will be shown that, being a relational structure, a CSM can be taken as the semantic specification for a hybrid multi-modal language, exploiting the features that such a logical approach provides (names for states and the possibility to embed satisfaction statements into the language). Then, since the aim of this paper is to look forward the integration with the MMASS model presented in Section 2, it has been chosen to present reasoning not by means of a calculus (although it can be proved that is possible to define a sound and complete tableaux based calculus for most significant classes of CSMs) but rather as model checking of hybrid formulas, and an example of how this could work will be given.

3.2 CSM: a Model for Commonsense Spatial Reasoning

A Commonsense Spatial Model $CSM = \langle P, R_S \rangle$ *is a relational structure, where* $P = \{p_1, ..., p_i\}$ *is a* finite *set of places, and* $R = \{R_1, ..., R_n\}$ *is a finite non-empty set of binary conceptual spatial relations, labeled by means of a set of labels* N.

A place can be any entity completely identifiable by a set of properties, as argued in the previous section. The set R can be any arbitrary set of binary CSRs, although some classes of relations, that are significant for a wide reasoning domains, will be analytically and formally characterized in the following paragraphs. This first lack of constraints on what can be taken as a CSR, may be seen as a weakness of the model, but is related to the main goals guiding this approach and to the fact that a morphological account of spatial relations is not the first object of a commonsense model of space as intended here. This mean that it is not possible (nor useful) to identify a minimal set of primitives relations as it has been done with other well known topological models (e.g. RCC calculus in [21], and spatial modal logic of [1]).

Nevertheless, as already mentioned, there are some significant classes of relations that play a special role in the definition of a commonsense model of space. In particular, a place can be "proximal to" another place, or it can be be "contained in", or it can be "oriented" with respect to another (distinct) place. The concepts of *proximity*, *containment*, and *orientation* identify three classes which, in their various declination, seem us to be archetypical in commonsense spatial reasoning and can be discussed according to the formal properties of the relations they group together.

Proximity. A first basic relational concept that is useful to define space concerns the possibility of reaching one place from another (in both physical and metaphorical sense). Two places are said to be proximal, in this sense, if it is possible to go from one to the other without passing through another place. A *proximity* relation R_P it is said to hold between two places p and q when the place q is directly reachable from place p. This relation can be modeled as a physical "adjacency" relation since it is *irreflexive* and *symmetric* (this is the type of relation exploited in most of discrete models of space based on both irregular and regular graphs, or cells grids). However, different criteria of proximity can be adopted (e.g. proximity relation as networking among devices in distributed systems).

Containment. Since places are arbitrary entities possibly with different shapes, dimensions and nature (e.g. a room and a printer are both places), a physical inclusion relation R_{IN} over places is needed in order to relate different types of places: an object may be in a room that may be in a building (where the object, the room and the building are interesting place of the same commonsense spatial model). The relation $R_{IN}(p, q)$ is interpreted as stating that the place q is *contained* in the place p; R_{IN} is a standard mereological relation: it is a partial order, that is, a *reflexive*, *antisymmetric* and *transitive* relation. Stronger antisymmetry, $i.e. \forall p, q(R_{IN}(p, q) \wedge R_{IN}(q, p) \rightarrow p = q)$, can be exploited to infer identity between two places for which is said that one is in another and vice versa.

Orientation. Finally, we need some relations to ensure *orientation* in space giving an account of entities' relative disposition: assuming that the concept of "reference point" is a rudimentary but fundamental way to start orienting into space. Assuming specific reference points consists in ordering entities with respect to these particular points, in such a way that every entity is put in relation with the reference point directly or indirectly. Whatever the contingent choice of the reference point, what is really important is the notion of order coming from its existence. A natural choice into a 2D space can be the exploitation of the four cardinal points, North, East, South, West, by means of the introduction of relations such as R_N, R_E, R_S, and R_W among places (where, $R_N(p, q)$ holds *iff* p is *at north of* q). The class of orientation relations consists of *strict partial orders* on the set of places that is, they are *irreflexive*, *asymmetric* and *transitive* relations; the order is "partial" because two places might be incomparable, and has always a greatest element, that is, a top element (e.g. the North for R_N). Some relations can be defined as the converse of other ones (e.g. R_S of R_N), and other non-primitive relations such as *at north-east of* (R_{NE}), can eventually be defined by means of usual set theoretic operators from the previous ones, e.g. $R_{NE} = R_N \cap R_E$.

These three relation classes play a fundamental role in commonsense spatial reasoning because, as far as a qualitative account of the things arrangement into space is concerned, the joint assumption of relations of the three classes above, although not provably exhaustive, provides a characterization of the environment, yet qualitative and rough, but which meets at least the following representational requirements:

• the definition of a basic graph relating places according to their reachability by means of proximity relations. This responds to the answer "where can I go from here?";

• a rough (qualitative) ordering of places into a 2Dd or 3D space by means of orientation relations (3D if a up-down order is added): this is important to someway reflect the idea of disposition of places and objects in space. Neither a grid nor a Cartesian reference system are necessarily exploited here, but the notion of disposition is traced back to the concept of *order*, and more precisely, to the projection of various orders on the place domains;

• the possibility of taking into account places of various types and size, representing different layers of abstraction by means of containment relations (a desk into a room, a room into a building).

Given the importance of those three classes in the emergence and definition of a Commonsense Model of space, we want to identify a special (more interesting) class of CSM that will be called *standard CSM* ($SCSM$), consisting of the set of CSMs where all the relations are of type *proximity*, *containment* and *orientation* and there is at least one relation for each category. Moreover, since the orientation relations are always defined with respect to a reference point, that is, the top of the respective order, the set of place must include a place for every orientation relation to be taken as the top element. Formally: *let assume that* $R_1^o, ..., R_n^o$, *is a set of orientation relations each one with its top element* top_i, $R_1^c, ..., R_n^c$ *is a set of containment relations, and* $R_1^p, ..., R_n^p$ *is a set of proximity relations. A standard commonsense spatial model* $SCSM$ *is a* CSM *where* $R = \{R_1^o, ..., R_n^o, R_1^c, ..., R_n^c, R_1^p, ..., R_n^p\}$ *and* $\{top_1, ..., top_n\} \in P$.

The model is basically defined, but besides the properties of each type of relation, it is possible to explicitly take into account also *cross properties* of the model that concern the relations among different CSRs and their interdependencies. Something has been said for orientation relations such as R_S and R_N, where converse definition is intuitive, but much more can be done, for example, if we want to exploit the inheritance of orientation relations through the containment relation (i.e. if a place p_0 is at west of a place p_1, every place contained in p_0 is at west of p_1). It will be shown in the next section that hybrid languages are very powerful in the specification of those cross properties.

3.3 Reasoning into Space: a Hybrid Logic Approach

Now the passage from the model to logic is straightforward: since every CSM is a relational structure, it can be naturally viewed as a kripkean semantic specification for a multi modal language. More generally, classes of CSMs defined by specific sets of relations (and by their properties) can be made correspond to "frames", whose relations define the meaning of the modal operators of the language.

Nevertheless, if modal languages are known to be suitable for reasoning about relational structures, and have been exploited for temporal, spatial and many other logics (see [9]), Hybrid Logic adds to the modal perspective features that are particularly useful with respect to our approach to commonsense spatial reasoning. In fact, Hybrid languages are modal languages that allow to express (in the language itself) sentences about satisfiability of formulas, that is to assert that a certain formula is satisfiable at a certain world (i.e. at a certain place in our framework). In other words, it is a formidable tool to reason about what is going on at a particular place and to reason about place equality (i.e. reasoning tasks that are not provided by basic modal logic).

Formally, this is achieved by adding to the language a specific *sort* of atomic formulas (i.e. "nominals") that are evaluated to be true at a single state of the model (whose valuation is a singleton in W, which is said to be the its *denotation*) and introducing a set of "satisfaction operators" $@_i$. Then semantics is given as usual for modal logic, and introducing the following truth condition statements for the set of satisfaction operators:

$$\mathcal{W}, w \models @_i\phi \text{ if and only if } \mathcal{W}, w' \models \phi,$$

where the place w' is the denotation of i, i.e. $V(i) = w'$.

	Property	CSR class	Definition
(*ref*)	*reflexivity*	C	$@_i\Diamond i$
(*irref*)	*irreflexivity*	P,O	$@_i\neg\Diamond i$
(*sym*)	*symmetry*	P	$@_i\Diamond\Diamond i$
(*asym*)	*asymmetry*	O	$@_i\neg\Diamond\Diamond i$
(*antisym*)	*antisymmetry*	C	$@_i\Box(\Diamond i \rightarrow i)$
(*trans*)	*transitivity*	O,C	$\Diamond\Diamond i \rightarrow \Diamond i$

Table 1. SCSM properties definability

One of the more interesting feature of Hybrid Logic is the opportunity of simply defining class of frames by means of pure formulas, that is, formulas that do not contain propositional variables. By means of those formulas, which allow to define properties not definable with plain modal logic, it is possible to completely define the class of frames corresponding to SCSMs. Table 1 contain the properties relevant to the SCMS relations, to which the following formulas asserting the existence of top element for the orientation relations and its unicity must be added:

$$@_i\Box\Diamond\top \leftrightarrow \neg\Diamond\top \qquad (ex)$$
$$@i\neg\Diamond\top \rightarrow @j\neg\Diamond\top \rightarrow @_i j \qquad (uni)$$

Now it is possible to exploit hybrid logic formulas to formally define a particular instance of a SCSM to be exploited with respect to MMASS. We want to grant flat orientation in space in four directions, adjacency among places (it is possible to reach a place from another one) and to model containment among places; with respect to the latter, in order to simplify reasoning, we introduce two containment relations defining the second as the inverse of the other one, so that it will be easy to reason along both the direction of the $\leq$ relation. We will call the following Basic Standard Commonsense Spatial Language and give its proper semantics [6].

The ***Basic Standard Commonsense Spatial Language*** *($SCMS^{basic}$) $\mathcal{L}^b$ is a hybrid language containing the modal operators $\Diamond_N$, $\Diamond_E$, $\Diamond_S$, $\Diamond_W$, $\Diamond_{IN}$, $\Diamond_{NI}$ and $\Diamond_A$, the respective boxes ($\Box_N$, and so on), and where*
$\{north, east, south, west\} \in NOM$.

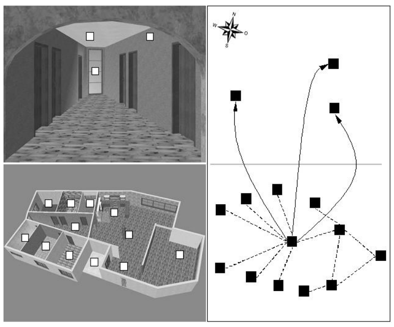


Figure 1. The emergence of a commonsense spatial model in the context of a monitored apartment. On the left a 3D model of the apartment and a cross-section of its corridor are presented. In the right side, the generation of the corresponding spatial model is represented: the nodes are the interesting places (rooms and sensors), while proximity and containment relations are represented by dashed and unbroken lines respectively. Orientation relations can be guessed but have been omitted for sake of clarity.

Formulas of $\mathcal{L}^b$ are interpeted over a $SCSM$: $\Diamond_{IN}$, $\Diamond_{NI}$ are interpreted over *containment* accessibility relations, $\Diamond_A$ over a *proximity* relation, and $\Diamond_N$, $\Diamond_E$, $\Diamond_S$, $\Diamond_W$ over *orientation* relations, whose top elements are respectively the denotation of *north, east, south, west*.

To have an overview about the meaning of interesting formulas consider that the set of pure defining the frames is given by the combination of those formulas with respect to the different operators:

$\Diamond_N$, $\Diamond_E$, $\Diamond_S$, $\Diamond_W$	*irref, asym, trans, ex, uni*
$\Diamond_{IN}$, $\Diamond_{NI}$	*ref, antisym, trans*
$\Diamond_A$	*irref, sym*

Moreover by means, again, of pure formulas it is possible to specify appropriately cross properties among CSRs. In particular, couples of orientation and containment relations are defined conversely, and orientation is inherited trough containment (i.e. if a place has a specific orientation with respect to another place, then every place contained in it inherits such an orientation):

$$@_i(\Box_N\Diamond_S i \wedge \Box_S\Diamond_N i)$$
$$@_i(\Box_E\Diamond_W i \wedge \Box_W\Diamond_E i)$$
$$@_i(\Box_{IN}\Diamond_{NI} i \wedge \Box_{NI}\Diamond_{IN} i)$$
$$\Diamond_\bigstar i \rightarrow \Box_{IN}\Diamond_\bigstar i \quad \text{where } \bigstar = (N|E|S|W)$$

Finally, interpretation of North is bound by the formula $@_{north}\neg\Diamond_N i$, and analogous formulas are introduced for the other top elements.

According to the aims of the modeled correlation task, a domain dependent set of properties can be chosen and represented in the formal language by means of a suitable set of symbols for propositional letters (e.g. the information "there is a man", coming from a local data processing, can be represented with a proposition "is_man", true or false at some place of the model).

The combination of the multimodal and hybrid expressiveness provides a powerful logical reasoning tool to shift perspective on a specific place by means of a $@_i$ operator, which allows checking properties holding over there. As an example, take a Smart Home scenario, as sketched in Figure 1, where different sensors are distributed into rooms. The system may have various task, among which, for instance intrusion detection and alarm managment; when a system devoted to intrusion detection need to query if "a glass is broken" at the place corresponding to the broken-glass sensor, the satisfiability of the formula $@_{window\_sensor} broken\_glass$ must be checked. Moreover, exploiting this operator, it is possible to define local access methods to explore the spatial model according to the other defined operators - e.g. checking the satisfiability of the formula $@_{kitchen} \Diamond_W \Diamond_{IN} smoke$ formally represents the verification, for the system, that "in" some room "at west of" the kitchen some "smoke" has been detected.

Here appears a new aspect that is crucial to the exploitation of a hybrid logic in a multi-agent system framework, an issue related to the double perspective over reasoning, both local and global, that this logic introduces. In modal logic reasoning and deduction start always from a given point of the model, i.e. from what is taken as the "current world". In terms of the interpretation of worlds as places, this means that reasoning is performed by a local perspective, and precisely, from the place in the environment taken as the current one. Since, according to the presented model, agents are situated in places, each agent can reason about context from its local perspective but exploiting a shared model of the environment. Taken an agent, checking the satisfiability of the formula $\Diamond_P (sensor \wedge broken\_glass)$ from its current place means to query if a broken glass has been detected by a sensor adjacent to it (an adjacent place on which $sensor$ and $broken\_glass$ are true). On the other hand, Hybrid Logic, still preserving the same local attitude to reasoning of classic modal logic, allows global queries such as $@_{window\_sensor} broken\_glass$. This, in fact means, that whatever is the agent on which reasoning is performed, the query regards a specific place/device, that is, the $window\_sensor$.

4 Integration of MMASS and CSM

4.1 Motivations

The previous Sections introduced two different models, one aimed at allowing the definition of multi-agent systems in which the spatial aspects of the environment are a crucial factor and another supplying concepts, abstractions and mechanisms supporting the representation and reasoning on facts related to spatial information and knowledge. The motivations to consider a possibile integration of the introduced models derives from experiences of applications of the MMASS to the development of simulations in different domains. The main consideration is that MMASS agents have a rather reactive flavor. The modelling activities generally provide the definition of a spatial structure, the modelling of types of signals which are exploited to define the interaction mechanisms among situated entities and then the specification of the behaviours related to agents which inhabit the system. Some of these entities can be immobile agents which are used to generate (possibly dynamical) properties of portions of the environment through the emission of fields. The latter is generally made up of a set of rules specifying that an action may take place according to certain conditions. The preconditions of these rules may be composed of elements of the model and thus do not allow a simple specification of complex expressions on properties of the environment. Moreover, the behaviour of agents depends on their state, which also determines the signals that they may perceive. State changes can thus represent the deliberation of a change in the goals of an agent. Once again, the preconditions of this kind of action cannot easily encapsulate a form of reasoning on beliefs and complex expressions on agent's context.

CSM has been introduced as a suitable structure on which represent and perform reasoning on facts related to spatial information and knowledge. The related hybrid language in fact allows to speak about the MMASS environment, to place information (as properties) at sites, and to reason about their meaningful correlation.

A way to integrate CSM and MMASS provides thus the possibility to exploit this language for the definition of parts of agents' state that are related to their beliefs on spatial elements of their context (including the presence of other agents). In this way it would be possible to specify preconditions to agents' actions that pertain complex expressions supported by this formalism. In this way, agents could be endowed with additional abstractions and concepts supporting a more complex form of reasoning about their behaviors (e.g. interaction with other agents, movement). This can be achieved by means of model checking techniques for formulas of the hybrid language introduced above, basically because of the homogeneity of the model of space embedded in MMASS and CSMs, the relational structures that provides the semantics to the hybrid language. Both MMASS and CSMs represents the environment through a graph-like model and this allows to draw a first mapping between the basic concepts of MMASS spatial representation and the CSMs relational structure. These concepts, together with their proper defining axioms, will become components of the agent state and they will be associated with domain-specific action selection strategies exploiting them.

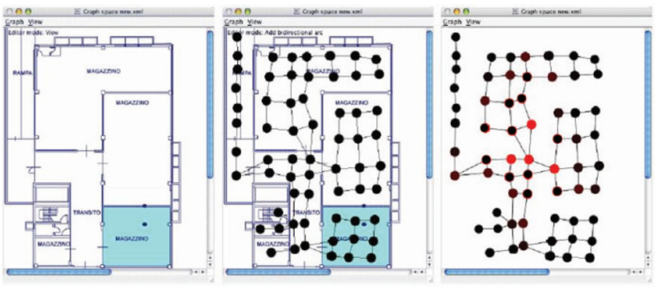


Figure 2. The figure show an example of how an agent environment can be modeled in MMASS: the nodes of the graph are sites, and the clearer points represent agents situated in them. In a CSM the sites become site-places and the agents agent-places contained in them. Site-places can be grouped together according to the room they are located in with respect to the map in the center.

4.2 Mapping

Firstly, the fundamental elements of the MMASS model, that is "agent" and "site", are translated in the CSM by means of the unique notion of "place": in CSM both sites and agents of MMASS become places. But, since places are (completely) identified by means of the set of properties holding on it, two properties such "site" and "agent" can be exploited also to qualify two first different types of places: places that are sites in MMASS and places that in MMASS are agents. For simplicity we will refer to those types of places respectively as *site-places* and *agent-places* (see fig. 2).

Secondly, however the adjacency relations among sites has been defined in MMASS (adjacency among sites can be defined following different criteria, e.g. arbitrarily, or based on Moore or Von-Neumann neighborhood on the sites sets), this relation is represented by means of a proximity relation R_{P_a} among site-places in the CSM; in fact, proximity relations are characterized exactly by the basic formal properties of adjacency relations. Finally, situatedness of agents, i.e. every agent is situated on a site, is represented by means of a containment relation R_{IN}, that is, agent-places are contained in site-places. In this way, the first mapping from a MMASS model leads to a CSM whose nodes are qualified by means of the distinction among agents and sites, and which includes a proximity relation holding among site-places according to adjacency of sites in MMASS and a containment relation among site-places and agent-places according to the disposition of MMASS agents into the sites' graph.

Now, this can be considered a first mapping between the two models, but how does it reflect on representation and reasoning as far as the Hybrid Language for CSM is concerned? Well, Hybrid Logic turns out to be extremely versatile in such contexts. What we need is to include in the language a set of nominals $i_1, i_2, ..., i_n$ to name both sites and agents we are interested in (for example, in the apartment of fig. 1, nominals can be $smoke\_sensor$, $camera\_1$, and so on). Then formulas such as $@_i site$ and $@_j agent$ allows to define the type of place, that is, they say that the place i is a site-place and the place j is an agent-place. An operator $\Diamond_{P_a}$ is interpreted over R_{P_a}, such that with a formula $@_i \Diamond_{P_a} j$ it is possible to express in the language that the place j is adjacent to the place i (remember that nominals are, syntactically, formulas and can be used like other formulas, with the difference that they are true always at a single state of the model). Finally, if an agent-place i is situated in a site-place j, this can be asserted by means of a formula $@_j\ (site \wedge \Diamond_{IN}\ (i \wedge agent))$, where $\Diamond_{IN}$ is the containment operator interpreted over the commonsense spatial relation R_{IN}.

Naturally, agents must know also important structural principles of MMASS which define the interactions among agents and sites, such as non-interpenetration and non-ubiquity (see section 2.1). Let i, i' be nominals for site-places and j,j' nominals for agent-places, these two principles can be modeled in the CS Hybrid Language with the following two formulas:

$$@_i \left(\Diamond_{IN} \left(j \wedge j'\right)\right) \rightarrow @_j j' \qquad (1)$$

$$@_j \left(\Diamond_{\overline{IN}} \left(i \wedge i'\right)\right) \rightarrow @_i i' \qquad (2)$$

The first formula says that if a site-place contains two different agent-places, these places are the same (i.e. the two nominals name the same place), and therefore guarantees the non-interpenetration principle. Conversely, non-ubiquity is expressed by the second formula, which asserts that an agent-place cannot be contained (i.e. situated) in different site-places, where the meaning of $\Diamond_{\overline{IN}}$ is defined as the inverse of $\Diamond_{IN}$.

Moreover, since a CSM has been defined as an open model, increasing the complexity of the hybrid language associated, it is possible to introduce new places, qualified by properties, and new orientation, containment and proximity relations whose meanings go beyond the original representational objectives in the MMASS model. In this way new, higher level information coming from abstractions of the agents' environment can be included in the commonsense spatial model exploited by deliberative agents.

As for the new places, it is interesting for the deliberative agents to be able to reason about the spatial disposition of places grouping together a large set of sites coming from the MMASS infrastructure, once that these sites are identified as semantic units (e.g. rooms, passageways, buildings, offices, halls, squares, etc.). For example, places as "kitchen room", "corridor" and so on (see fig. 1), can be used to abstract a higher level representation with respect to the MMASS spatial model and named in the language with specific nominals; the site-places in CSM are linked to those new places by a containment relation, so that, e.g. a place "kitchen" contain site-places that on their turn may contain an agent-place. As for the extension of the set of CSRs, orientation relations turn out to be particularly important to distinguish among adjacent places (e.g. the formula $@_{kitchen} \Diamond_W corridor$ say that there exists a place "corridor" at west of the room "kitchen"). Orientation relations are inherited through the containment relation as stated by the following formula:

$$@_i \left(\Diamond_* i' \wedge \Diamond_{IN} j\right) \rightarrow @_j \Diamond_* i', \qquad (3)$$

where $* \in \{N, E, S, W\}$ and i, i', j are nominals for places. Moreover, observe that new places and relations introduced in CSM admits the possibility, proper of MMASS, of representing and connecting different layers according to different levels of abstraction (see Fig. 2).

4.3 Reasoning

Behavioral characteristics of deliberative agents strongly depend on their inferential capabilities. Actions selection strategies and sensitivity threshold to fields are influenced by the present status of the agent and, during is life, the status of the agent is in fact influenced by its spatial model-based reasoning activity. In this framework, MMASS agents that reasons upon the spatial disposition of the entities populating the environment, perform their reasoning by means of model checkers for hybrid formulas. The model checker checks the satisfiability of the truth conditions of multi-modal propositional formulas according to a given model (a Kripke structure with a CSM frame *and* a valuation function).

The formula evaluation process consists in searching through the graph of sites (that represents the agents' environment) for a node at a given proposition holds. Since the number of sites in the graph is finite, the evaluation process will have to explore only a finite number of graph nodes. Since the model checking technique explores the graph by following the semantics associated to the modal operators of the language, one of the outcomes of the formula evaluation process consists in the definition of specific paths of nodes throughout the graph. Any vector of nodes, ending with a site at which the proposition of interest holds, may become a movement-action plan for the agent in MMASS. The complete cycle, involving the spatial reasoning performed on agents' knowledge bases and the MMASS mechanisms for managing agents' actions, can be depicted as follows:

Step 1. The MMASS based system provides a goal to agents, such as the sensitiveness to a given field type associated to a certain relevant such as the exit of a building in a evacuation context (the same goal to all agents or, in a more complex situation, different goals for each agent).

Step 2. MMASS knows the location of each agent at an instant of time, and it is possible to map this location to a hybrid logic place (for instance through a mapping between specific field types and places).

Step 3. According to the location of an agent, an algorithm translates into a hybrid logic formula the goal of the agent (e.g. looking for the exit of the building); the verification of the satisfiability of that formula starts from the site in which the agent is located.

Step 4. The satisfiability process decomposes the formula, making explicit the intermediate places (and thus field types) the agent have to reach in order to achieve its final goal.

Step 5. The resulting graph of places, that are needed to satisfies the formula, is thus transmitted to the MMASS system that will take care of the movement of the agent.

Now, consider an example dealing with the "emit" action, in the context of the Smart Home case depicted in fig. 1. In the MMASS computational framework, a graph of sites represents the environment in such a way that the several devices spread over it are modeled as situated agents (agents can be: a sensor, an access point, a PDA, and so on). Given to a specific agent, e.g. the monitoring and control component of the system, an explicit representation of the environment through a CSM, it is possible to add logic-based preconditions to the agent's action selection strategy. Suppose that an alarm must be diffused by this agent (through a field emission) when he infers that some smoke has been detected by a "sensor contained in a site at west of the kitchen", the precondition of this "emit" action can be represented exploiting the higher level representation of the environment by the hybrid formula:

$$@_{kitchen}\Diamond_W \Diamond_{IN} smoke \tag{4}$$

Given a model $\mathcal{W}'$, the model checker verifies the truth conditions of the above formula exploring the model through the following steps:

$\models @_{kitchen}\Diamond_W \Diamond_{IN} smoke$	iff	there exists a place $w = kitchen$, s.t.
$\models_w \Diamond_W \Diamond_{IN} smoke$	iff	there exists a place w', with $wR_W w'$, and
$\models_{w'} \Diamond_{IN} smoke$	iff	there exists a place w'', with $w'R_{IN}w''$, and
$\models_{w''} smoke$		★

It is relevant to observe that the finiteness that always characterizes the CSM model (namely, the set of places and relations is finite), guarantees that the verification algorithm of the formulas satisfiability, implemented in the model checker, terminates.

5 Concluding Remarks

This paper presented two different models, and more precisely the MMASS which supports the definition of complex MASs in which the environment, and especially its spatial aspects, can be explicitly represented and exploited by interaction mechanisms, and CSM which represents a structure on which represent and perform reasoning on facts related to spatial information and knowledge. The main aim of the paper was to propose an integration of the two models aimed at supplying more expressive abstractions supporting the behavioural specification of MMASS agents. In particular, CSM represents a suitable formalism to express complex conditions involving information related to facts representing properties of elements of a spatial abstraction. These conditions can be exploited for the definition of preconditions to MMASS actions. The next steps of this research are aimed at the implementation of the model checker for the SCSM language into the MMASS platform and at the design of applications exploiting the proposed integration in the context of ubiquitous systems and supporting the simulation of crowd behaviours in realistic environments (e.g. shopping centres).

Interesting suggestions for future developments, especially involving the hybrid logic introduced in the paper and the supported spatial reasoning capabilities, come from the works on the ontological enhancement of agent knowledge bases [18, 11]. In fact, ontological knowledge provides information that are necessary to qualify the environment, therefore increasing the reasoning capabilities of agents acting and interacting in it. Ontological knowledge can be naturally represented within the hybrid multi-modal paradigm by means of sets of propositional symbols holding at specific places of the model.

Actually, there are many domains in which time dimension is crucial and a very interesting problem for further formal and theoretical work is how to consider time and dynamism integrated with CSM. On one hand, in fact, considering the dynamical evolution of a MMASS system, the explicit spatial reasoning may need to relate facts true at different places at different time (properties holding over a place change in time). On the other hand, in domains characterized by the presence of wireless technologies, interesting places, properties holding over them and the relations' extension may change, since new interesting places can be discovered (e.g a mobile agent is identified as a place) and known places can move.

REFERENCES

[1] M. Aiello and J. Van Benthem, 'A modal walk through space', *Journal of Applied Non-Classical Logics*, **12**(3-4), 319–363, (2002).

[2] S. Bandini, S. Manzoni, and Carla Simone, 'Heterogeneous agents situated in heterogeneous spaces.', *Applied Artificial Intelligence*, **16**(9-10), 831–852, (2002).

[3] S. Bandini, S. Manzoni, and G. Vizzari, 'Multi–agent approach to localization problems: the case of multilayered multi agent situated system', *Web Intelligence and Agent Systems*, **2**(3), 155–166, (2004).

[4] S. Bandini, S. Manzoni, and G. Vizzari, 'Situated cellular agents: a model to simulate crowding dynamics', *IEICE Transactions on Information and Systems: Special Issues on Cellular Automata*, **E87-D**(3), 669–676, (2004).

[5] S. Bandini, S. Manzoni, and G. Vizzari, 'A spatially dependant communication model for ubiquitous systems', in *The First International Workshop on Environments for Multiagent Systems*, volume 3374 of *LNCS*. Springer-Verlag, (2005).

[6] S. Bandini, A. Mosca, and M. Palmonari, 'Common-sense spatial reasoning for information correlation in pervasive computing', *Applied Artificial Intelligence*, **21**, 405–425, (2007).

[7] S. Bandini, A. Mosca, and M. Palmonari, 'Commonsense spatial reasoning for context–aware pervasive systems', in *Proceedings International Workshop on Location- and Context-Awareness (LOCA2005)*. Springer-Verlag, (To appear).

[8] Stefania Bandini, Mizar Luca Federici, and Giuseppe Vizzari, 'Situated cellular agents approach to crowd modeling and simulation', *Cybernetics and Systems*, **38**(7), 729–753, (2007).

[9] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, Cambridge University Press, 2000.

[10] Paul Hsueh-Min Chang, Kuang-Tai Chen, Yu-Hung Chien, Edward Kao, and Von-Wun Soo, 'From reality to mind: A cognitive middle layer of environment concepts for believable agents.', in *Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, Revised Selected Papers*, volume 3374 of *Lecture Notes in Computer Science*, pp. 57–73. Springer, (2005).

[11] K. L. Clark and F. G. McCabe, 'Ontology schema for an agent belief store', *Int. J. Hum.-Comput. Stud.*, **65**(7), 640–658, (2007).

[12] E. Davis, *Representations of commonsense knowledge*, Morgan Kaufmann Publishers, 1990.

[13] Jacques Ferber, *Multi–Agent Systems*, Addison–Wesley, 1999.

[14] S. Gopal, R.L. Klatzky, and T.R. Smith, 'Navigator: A psychologically based model of environmental learning through navigation', *Journal of Environmental Psychology*, **9**, 309–331, (1989).

[15] B. Kuipers, 'Modelling spatial knowledge', *Cognitive Science*, **2**, 129–154, (1978).

[16] D. Leisler and A. Zilbershatz, 'The traveller: A computational model of spatial network learning', *Environment and Behaviour*, **21**(4), 435–463, (1989).

[17] Marco Mamei, Letizia Leonardi, and Franco Zambonelli, 'A physically grounded approach to coordinate movements in a team', in *Proceedings of the 1st International Workshop Mobile Teamwork*, pp. 373–378. IEEE Computer Society, (2002).

[18] Álvaro F. Moreira, Renata Vieira, Rafael H. Bordini, and Jomi Fred Hübner, 'Agent-oriented programming with underlying ontological reasoning', in *DALT*, eds., Matteo Baldoni, Ulle Endriss, Andrea Omicini, and Paolo Torroni, volume 3904 of *Lecture Notes in Computer Science*, pp. 155–170. Springer, (2005).

[19] H. Van Dyke Parunak, 'Go to the ant: Engineering principles from natural multi-agent systems', *Annals of Operations Research*, **75**, 69–101, (1997).

[20] H. Van Dyke Parunak and Sven Brueckner, 'Ant-like missionaries and cannibals: synthetic pheromones for distributed motion control.', in *Agents*, pp. 467–474, (2000).

[21] D. A. Randell, Z. Cui, and A. G. Cohn, 'A spatial logic based on regions and connection', in *Proc. 3rd Int. Conf. on Knowledge Representation

and Reasoning, pp. 165–176, San Mateo, CA, (1992). Morgan Kaufmann.

[22] Anand S. Rao and Michael P. Georgeff, ‘Bdi agents: From theory to practice.’, in *ICMAS*, pp. 312–319. The MIT Press, (1995).

[23] D. E. Steiner, H. Haugeneder, and D.E. Mahling, ‘Collaboration of knowledge bases via knowledge based collaboration’, in *International Working Conference on Cooperating Knowledge Based Systems*, pp. 113–133. Springer-Verlag, (1991).

Don’t Give Yourself Away: Cooperation Revisited

Anton Nijholt[1]

Abstract. In the future, sensor-equipped environments will be able to detect, interpret and anticipate our intentions and feelings. While on one hand this allows more natural interaction between humans and human activity supporting intelligent environments, on the other hand it allows these environments to collect more information about the user than he or she finds desirable for a satisfactory interaction. That is, there are many human-human interaction situations where it is quite acceptable or even necessary that part of the intentions and feelings of one conversational partner remains hidden for the other. We will discuss research on ambient intelligence and human-computer interaction that allows us to introduce and discuss this problem and we illustrate our viewpoints with examples from our own research on virtual humans interacting with human partners that act, behave, and perform in environments equipped with sensors that capture and interpret human behaviour.

1 INTRODUCTION

Most of our research in human-computer interaction assumes that humans and computers cooperate. And although there is research on adaptive interfaces, most of the time the user has to adapt to the interface by using rather unnatural devices, follow interaction protocols, speak clearly, etcetera. Here we explore human-computer interaction where there is not necessarily cooperation and where it may be in the interest of the user to hide his intentions or feelings. When we talk about interaction then we don’t limit ourselves to verbal interaction. On the contrary, in what follows we assume situations and environments where all modalities that can be displayed (movements of body parts, posture and gestures, facial gestures, speech, gaze, (neuro-) physiological) can be observed by the interacting partners.

2 NO DESIRE TO BE FORTHCOMING OR COOPERATIVE

People often hide their feelings, they often hide their thoughts, and they often hide information. People often behave differently depending on when they are alone or when others are around. People sometimes want to hide from others; they are not always in need of an audience, bystanders or partners.

People have their interests and preferences. Depending on them, and their personality and their mood, they voluntary or involuntary give away part of themselves during interactions. People do not always want to be forthcoming. Moreover, they play roles. Implicit or explicit decisions are made about the roles they want to play and what they want to disclose, where, when, and how. This is also known as privacy. They also make decisions how much effort they will make in understanding a conversational partner and his or her interactional goals. Also, too much interest from others in our motivations is not appreciated. We don’t want other people to read our mind. We are not always interested in reading other people’s mind.

Neither is it always in our interest to be cooperative. However, in general, being cooperative, just as being polite, can sometimes help us to get closer to our interactional goal. In a conversation we can flatter our conversational partner, we can purposely misunderstand our partner in order to be able to make a humorous remark, and we can play the devil’s advocate, and nevertheless be cooperative. We play along with the rules of a conversation or a negotiation and therefore we are cooperative despite possible elements of competitiveness. In these situations Grice’s maxims on cooperation, i.e. assumptions a listener is supposed to have about the interaction behaviour of a speaker, seem to be violated, but the relevance of the behaviour can be explained from a pragmatic, conversational point of view, rather than from a sentence level point of view. Conversational partners can achieve their goals although they can have competitive interests. To achieve these goals it is acceptable that people hide their feelings and intentions. Moreover, it is quite acceptable that they tell lies.

Clearly, lies don’t follow Grice’s maxims. People don’t follow Grice’s maxims, since all people sometimes lie in everyday conversations. They say things they don't mean, they tell self-oriented lies, i.e., lies for one’s own benefit, or other-oriented lies, i.e., for the benefit of others. Social lies are meant to benefit relationships. Lies act as a social lubricant [1]. We don't always want to speak the truth; we don't always want to hear the truth. And sometimes we don’t want to find out the truth. Lies are in the interest of both conversational partners in an interaction. In addition to telling the truth, during a row people will also exaggerate or say things they don’t mean. Lies can be nonverbal. We can nonverbally feign surprise or sincerity. We can pretend to be happy by our facial expression; we can pretend to be rich by our clothes.

Interactions can be completely nonverbal or nonverbal supported by speech. Consider for example, two partners coordinating their behavior during dancing, students coordinating their movements with a fitness trainer or a sports instructor, and a conductor conducting an orchestra. Clearly, in sports (e.g., baseball, soccer, tennis, …) and in games misleading your opponent by feigning movements, introducing sudden interruptions of movements or changes in behavior is essential and is part of the entertainment or sports experience. Nonverbal humor is an example of friendly misleading your partner, e.g. during disco dancing by feigning certain movements.

[1] University of Twente, Human Media Interaction, PO Box 217, 7500 AE, Enschede, the Netherlands. anijholt@cs.utwente.nl

In all these observations the nonverbal characteristics of the interaction are extremely important. Even during conversations these characteristics can say more about what people mean or want others to believe than the content of the verbal utterances. People can read these nonverbal characteristics and they can be misled by these nonverbal characteristics in their interactions.

Obviously, there is not necessarily a balance between capabilities of conversational partners. Partners differ in background, knowledge, attitudes and personality. A partner can be more determined to reach a certain goal, a partner can have more social intelligence and be able to read the mind of its human opponent better than he or she is able to do. Not all partners in interactions have equal means.

3 DISAPPEARING COMPUTERS AND INTERFACES

Human-computer interaction is about designing interfaces between humans and computers. Before there were Personal Computers for most of the computer users the interface consisted of a counter where you could deliver your program punched on tape or cards. After that end users were allowed real-time remote access to computers using terminals that allowed the composing and editing of programs. Personal Computers provided end users with facilities to interact with software designed by software companies or by themselves. User interface software mediated between users and application and system software. Graphical user interfaces aimed at efficient and user-friendly interaction. Interface technologies now include speech and language input, haptic input, and vision input. Moreover, in addition to professional applications where efficiency is important and necessary, home and recreational computer use became extremely important and these applications require interfaces where there is a user access layer where user friendliness, ease of learning, adaptiveness, and fun to use are the main design issues, rather than efficiency considerations that appear in levels below.

As mentioned, interface technologies now include speech and language input, haptic input, and vision input. But there is more. Since we can have sensors embedded in the environment, including walls, furniture, devices, robots and pets, now the environment has become intelligent and it can perform not only reactive, but also pro-active behaviour, trying to anticipate what the inhabitant is doing and doing this by perceiving activities and all kinds of verbal and nonverbal behaviour. Embedded sensors include cameras, microphones, location and movement sensors, and sensors that collect and distinguish various types of physiological information and brain activity patterns. Information about the behaviour of the inhabitants and their implicit and explicit addressing of the environment can be fused and interpreted in order to support the inhabitants by providing appropriate feedback.

This research, in particular when research activities into social and intelligent interfaces are included, has become known as ambient intelligence research. Well-known is also Mark Weiser's vision of disappearing computers ("The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.") [2] and the associated question how we can design implicit and explicit interaction, with multiple human inhabitants, for sensor-based interfaces [3]. In our view [4] in these environments humanoids and pet-like devices can play a useful role in observing inhabitants and interacting with them. Agent-modelled virtual humans, (mobile) robots, pets (virtual or robotic animals) can have specific human-oriented tasks in smart environments (e.g., be a friend, assist in cooking, take care of house security, retrieve information, assist in health-care and fitness, be an opponent in games or sports), and they can represent human beings (e.g., family members that are away from home but have there whereabouts or activities visualized). They can also communicate with each other, distributing, sharing and integrating their experiences, and learning about the inhabitants and visitors of these environments, making them transparent, independent of the role they play. Being able to capture and interprete what is going on Capturing and interpreting of events and activities, e.g. in home environments, allows future retrieval and replay of events and experiences in, for instance, 3D virtual environments [5].

4 NOT GIVING AWAY YOUR INTENTIONS OR FEELINGS

The main topic of discussion in this paper is not about privacy or privacy protection. But clearly, when we talk about not giving away intentions and feelings, there is the underlying assumption that you are acting and behaving in a world (ambient intelligence environment, smart environment) that is inhabited by agents that perceive your acts and behaviour and that may profit from the knowledge that is obtained in that way. And, they may have goals that do not necessarily match yours, maybe on the contrary. Hence, such agents may collect information that threatens your privacy by using this information without you being aware of it, without your consent, and sometimes against you. We can not expect that we can always choose our own agents. The providers of our smart environments will make choices, they have commercial interests and they have to follow political burps and fluctuating political and societal demands. Where can we hide from those who provide our environments with sensors and communicating intelligent agents that allow the environment to pretend social and intelligent awareness while collecting information during the interactions with us [6]?

Their agents will have their own interests, or rather their owners' interests, and therefore they will persuade us to adapt to their goals. As mentioned in [7], there will even be situations where our virtual personal assistant will take a decision to deceive us (in our 'own interest', of course) and also situations where we will want to deceive our personal assistant.

In our research we don't look into details of these issues. Rather we look at human behaviour during natural conversations and other interactions and the reasons to hide information, i.e., not display ourselves, not to be forthcoming or, even, wanting to mislead our (artificial) interaction partner and provide him or her with information that is not necessarily complete or true.

In future ambient intelligence environments, are we still able to provide our conversational partners with incomplete and sometimes wrong information about ourselves, our intentions and our feelings just as we are able to do and are used to do, and probably for good reasons, in real-life situations nowadays? This question can be asked for real-time face-to (virtual) face conversational interactions, but there are also many other situations where it is rather natural not to show everything we know or everything we feel. In our research on continuous interaction modelling (in contrast to 'turn-taking' interaction [8]) we designed and implemented applications where it turned out that humans in their interactions with embodied partners felt that there sometimes were advantages in not displaying their intentions and feelings, and, also the other way around. This became clear in our research on a so-called Sensitive Artificial Listener (SAL), developed in the framework of an EU FP6 Network of Excellence on the role of emotions in the interface, in which we participated [9], in our research on an interactive virtual dancer [10], an interactive virtual conductor [11], an interactive virtual fitness trainer [12], and in our research on an educational virtual environment for nurse education [13].

In these applications both the nonverbal interaction behaviour and the fact that during interactions all conversational partners continuously display nonverbal interaction behaviour [14], made clear that continuously decisions are being made about what a human partner would like to become displayed to a virtual interactional partner, and the other way around. Examples that emerged in our applications are: using humour to temporarily mislead a conversational partner, not being sincere by feigning interest in a conversation, not yet wanting to show your fatigue to your fitness trainer or colleagues, feigning movements during a dance interaction, and feigning movements in a virtual reality entertainment game.

5 HUMAN COMPUTING TECHNOLOGIES

Many authors have discussed smart environments, ambient intelligence, ubiquitous computing or pervasive computing. Mostly their starting point is the technology that makes it possible to embed environments with sensors, intelligent sensors, and communicating sensors. Our starting point is the support of human beings in their activities in smart environments. That is, rather than in traditional computer science where the assumption is that human beings should add to the tasks that a system has to perform, we investigate how sensor-equipped environments can support human beings in their daily home and professional activities, how they can contribute to their well-being, and how they can contribute to their curiosity to explore new areas of interest. For that reason we need to look at anticipatory user interfaces that should be human-centred and should be built for humans based on human models.

Our natural environments, whether they are home, office, or public space environments, become smart. They are being equipped with sensors, where the sensors themselves can be intelligent, but, more importantly, where the information that is obtained from various kinds of sensors is fused and interpreted, allowing relevant feedback generation from the environment and its devices to the inhabitants of the environment. Tangible objects, robots, virtual pets, furniture, walls, doors, virtual humans displayed on screens, etc., may re-actively and pro-actively provide support to real humans in their daily professional, home, and recreational activities. While doing this, these environments need to collect information about the user: his activities, his intentions, his moods, and his emotions. We discuss three viewpoints related to the ambient intelligence approach to computing and supporting inhabitants of ambient intelligence environments.

The first viewpoint in our research concerns the level of sophistication at which the environment understands what is going on and is able to provide support [15]. Depending on this level of sophistication models are needed that range from simple rules, such as, turn the light on when someone enters the room, to cognitive user models that are used by the environment to understand, e.g., why a person is choosing a particular chair around a table in a meeting room.

The second viewpoint concerns the way the environment and its devices are asked to use the embedded knowledge or decide to use this embedded knowledge in support of the activities of the environment's inhabitants or visitors. For example, in [16] we find an interaction space divided by two axes. One axis ranges from re-active to pro-active behavior, while the other axis covers the range from low to high level attention of these sensor-equipped smart environments. The main message here is that we can design intelligence in the interface and that due to this intelligence, in particular intelligence that is fed from observations from the environment, the interface can evoke behavior that (1) hides processing from the user, (2) can provide feedback to questions or actions requiring feedback, (3) can anticipate user's actions, and (4) can enter and entertain the interaction with other users.

The third viewpoint is related to the research methodology. Our research starts with observing how people behave and interact. For example, in the AMI and AMIDA project [15] we have a large collection of data concerning people interacting with each other. This data is annotated using annotation tools and annotation schemes. Tools and schemes are not perfect. However, machine learning technologies can be used to improve in an iterative way, analysis and modelling of (multi-party) interaction.

6 (NON-) COOPERATIVE BEHAVIOR BY USERS, PARTNERS, AND OPPONENTS

Our research is on natural cooperative and uncooperative behaviour. Clearly, we look at Grice's maxims, but we also distinguish situations where people may prefer not too be fully informative, not to display all relevant information, and maybe even prefer to mislead their interaction partner. It is not always in your interest to open yourself to a conversational partner. Moreover, interactions can become much more interesting and useful when such conversational rules are neglected or violated. This is certainly the case in applications where your interests forces you to disagree with your partner (or rather opponent), for example in discussions, games, or sports. Clearly, we are not talking about verbal interaction only. In our research we include nonverbal interaction and also all kinds of other activities that can be perceived by our interaction partners (or opponents).

It should be mentioned that there is a friction that emerges when on the one hand our smart environments and processing technologies not only allow, but also invite natural interaction behaviour, while on the other hand the processing technologies become able to extract more information about our intentions and feelings from this natural interaction behaviour than we would like to become known in a natural human-human interaction situation. How to deal with partners that have not necessarily been designed to help us, how to deal with partners, e.g. in games and sports that are opponents rather than friends? In the remaindrr of this paper we will in particular look at entertainment and sports applications. Therefore, in the next section, we will look at such applications and introduce the so-called exertion interfaces.

7 DANCES, SPORTS, GAMES, FITNESS

Entertainment, health, sports, and leisure applications using information and communication technology often require and encourage physical body movements and often applications are designed for that reason. In our research we look at bodily and gestural interaction with game and leisure environments that are equipped with sensors (cameras, microphones, touch, and proximity sensors) and application-dependent intelligence (allowing reactive and proactive activity). Interpretation of the bodily interaction, requiring domain-dependent artificial intelligence, needs to be done by the environment and the agents that maintain the interaction with the human partner. In the display of reactive and pro-active activity embodied virtual agents play an important role. Virtual agents can play the role of teacher, coach, partner or buddy. One underlying assumption is that emphasis on activities in which the experience rather than the result will be used to guide the design of social and intelligent systems that will become part of ambient intelligence home environments [5].

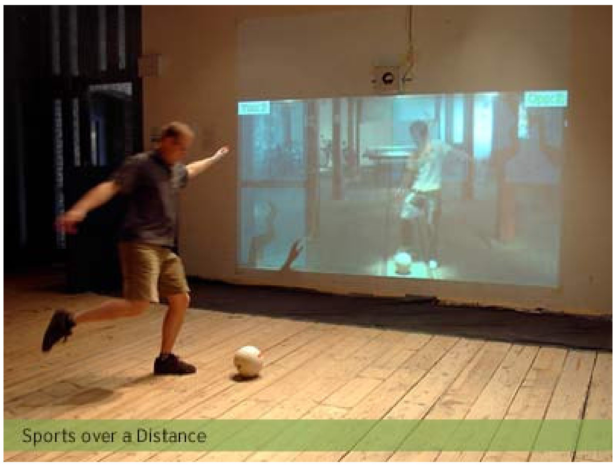


Figure 1. Sports over a Distance

Bodily activity, to be captured by cameras, microphones, pressure and location sensors, has been considered for many applications related to sports, games, entertainment, fitness, and education. Hence, there is a virtual therapist that helps patients to recover from injuries [17], an exercise bicycle as user interface to a virtual environment [18], a Tai Chi training master [19] and a shadow boxer [20] that acts as a fitness environment to help prevent neck and shoulder pain, or a Kick Ass Kung-fu system where the children use Kung-fu to fight virtual enemies displayed on a screen [21].

Interfaces to such environments have been called physical interfaces or exertion interfaces. Especially for the latter it is assumed that the interaction requires intense physical effort, for example, repeatedly shooting balls against a wall. This latter example has been realized in the “Sports over a Distance” project at the Media Lab in Dublin [22]. In their exertion interface users are connected through a video conference screen. For each of them the remote player is displayed on the screen. The screen is divided into blocks and the players have to strike the blocks with a regular soccer ball in order to score (see Figure 1). Blocks can ‘crack’, ‘break’, and ‘disappear’. Players see the results of the other player and can try to break a block that has been cracked by the other player. A really hard strike will break a block at once. Hence, in the game there is a combination of tactics and intense physical effort. More recently “Airhockey over a Distance” has been introduced. In this approach not only airhockey tables are networked and augmented with video conferencing, but there is also a physical puck that can be shot back and forth between the two connected locations by measuring the intensity of hitting the ‘wall’ and ‘firing’ a puck at the remote location from the position where it hit the wall.

In our HMI Lab we have designed three applications in which our ideas about nonverbal and bodily interaction have been implemented. The applications are illustrated in Figure 2. The implementations are there, but they are certainly not final. We looked at the design, implementation and evaluation of a virtual dancer that invites a visitor to her environment to dance with her, a conductor that guides musicians in its environment to play according the score designed by a composer, and a virtual trainer (e.g. in the role of fitness trainer or physiotherapist) that knows about exercises that need to be performed by a user or patient. In all these applications there is a continuous interaction between embodied agent and its human partner. Moreover, rather than have the more traditional verbal interaction supported by nonverbal communication, here the main interaction that takes place is nonverbal, and speech and language, when present at all, take the supporting role. External signals like music being played can also have a role in addition to the multimodal communication.

It should be mentioned that industry (Sony, Microsoft, Nintendo) have become aware of applications where users move away from keyboard, mouse and joystick. The dance pad, the DanceDanceRevolution (DDR) games and the DDR tournaments are examples, but so are the Sony EyeToy games that use computer vision input and the Nintendo Wii motion-sensitive

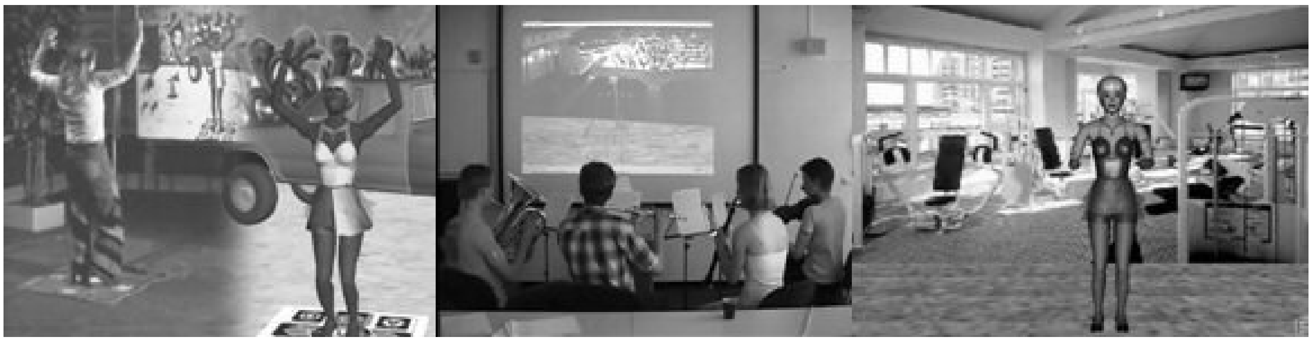


Figure 2. Virtual dancer, virtual conductor and virtual fitness trainer

input device. And, finally, Microsoft has investigated the use of a dance pad to allow a user to issue commands to his or her email program [23].

8 PLAYING WITH BEHAVIORAL INFORMATION

In ambient intelligence environments we have the technology to capture human behavior in everyday life. In our ambient entertainment view the same technology is available and we can either assume that a particular user or visitor of our ambient entertainment environment already carries a user profile that has been generated from the user's behavior in the past, or we can assume that during a possibly playful interaction with the environment a profile can be obtained and can be used by the environment to adapt to the user's characteristics (for example, personality, preferences, mood and capabilities).

What can we learn about the user when we can observe his or her behavior during some period of time? What can we learn from behavioral information captured by cameras, microphones and other types of sensors? In [24] results are reported from short observations of expressive behavior. Observations include the assessment of relationships, distinguishing anxious and depressed people from normal people, predicting a judges' expectations for a trial outcome, determining political views of television newscasters, et cetera. Personality judgments from 'thin slices of behavior' and their accuracy are also discussed in [25].

An example where real-time behavioral analysis is done by a computer can be found in [26]. In their approach a participant is invited in front of a video camera for about 30 seconds. At the end of this period a personality profile for the earlier mentioned Big Five personality traits will be generated.

Taking into Account Involuntary User Responses

In the examples mentioned in section 7 we have bodily interaction with the computer system. Input to an entertainment environment can be based on conscious decisions made by the human. This is usually the case when keyboard, mouse or joystick is used. In the examples we have body and hand gestures, changes of position, etc., to 'control' the system and to perform a certain entertaining task.

Information collected in a user profile, possibly obtained with the methods discussed in section 3 and 4, can be used to adapt an entertaining activity in advance to a particular user and during the activity it helps to anticipate and interpret the user's actions in the environment.

Behavioral signals and patterns during activities provide (additional) information about the tasks that a user wants to perform, the way they should be performed and the user's appreciation of task, performance, and context. Sensing and understanding these signals is an important issue in 'human computing' [3] and it makes human computing an important area of research for entertainment computing. This kind of input is not always consciously provided by a user and is sometimes beyond the control of the user. Behavioral signals also provide information about the affective state of the user and this information is needed to adapt the environment (more or less control by the user, other challenges, etc.) to the user.

More information about the affective state of the user of an entertainment environment can be obtained by collecting and interpreting information obtained from measuring physiological processes and brain activity. Physiological cues are obtained from, for example, respiration, heart rate, pulse, skin temperature and conductance, perspiration, muscle action potentials and blood pressure [27]. Unfortunately, this information can mostly not be obtained unobtrusively.

Finally, we should mention measured brain activity. Again, measuring brain activity, e.g. by using an EEG cap, can provide information about the affective state of the user (frustration, engagement, etc.) and this can be used to dynamically adapt the interface to the user and provide tailored feedback.

User Control of 'Involuntary' Responses

HAL: I'm afraid. I'm afraid, Dave. Dave, my mind is going. I can feel it. I can feel it. My mind is going. There is no question about it. I can feel it. I can feel it. I can feel it. I'm a... fraid.

From: A Space Odyssey, Stanley Kubrick, 1968

Playing against a computer is not fair. The computer knows about our affective state and can decide to communicate this information to our (virtual) opponents or collaborators in the environment who can use it to their advantage. On the other hand, apart from adapting the environment, the computer can also make the human player aware of his or her affective state so that he or she can make an attempt to control it since it can decrease own performance and give away unwanted information to other players in the game.

In games and sports opponents can be misled. We can as well try to mislead or tease our virtual and human partners who perform in a computer-controlled environment. One step further is that we have entertainment games where misleading the computer is essential part of the game. A simple example is playing soccer against a humanoid robot and the robot's aim is to win rather than to offer its human partner an enjoyable experience. In such a situation misleading means for example making feints. But also, for example, trying to look more tired than we really are and all other kinds of misleading behavior. In our virtual dancer installation (section) it happens that human dancers try to tease the virtual dancer by doing something much unexpected and see how she reacts. In other environments we may want to hide our intentions from the computer by controlling our facial expressions (e.g., in a poker game with a computer that can observe us). That is, once we know that our non-human opponent is receptive for our behavioral, physiological or even brain processes, we can try to cheat in order to obtain more satsfaction from the entertainment game. Although game research in this direction is rare, it is well-known that people can learn to control, up to a certain level, these processes. Research and development in brain-computer interfacing and its medical applications makes clear that interesting new types of entertainment will become available in the future [28, 29].

9 CONCLUSIONS

In ambient intelligence environments the environment learns about the user. The environments will be inhabited by, among others, virtual humans and human-lime robots that are there to support us, but also to support other humans making use of these environments. Human computing technology allows the environment to extract information about the user from his behavior and his activities, including his interaction behavior. Our assumption is that for maintaining natural human behavior in such environments we should be able to hide information from the artificial humans in these environments, we should be allowed to lie, and we should have the opportunity to mislead our artificial partners. This is particular true in situations where we play games or do sports with our artificial partners as we illustrated with some examples from our own research. In that case we look at the computer as our opponent, in addition to being a provider of enjoyable experiences. Rather than providing the computer with information about ourselves we then prefer to mislead the computer and hide information about our affective state or even control and manipulate our behavioral, physiological and brain processes so that we consciously can provide the computer with misinformation in order to become the 'winner' in smart entertainment environments.

Acknowledgements
This work has been supported by funding from the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

REFERENCES

[1] A. Vrij. Detecting Lies and Deceit: Pitfalls and Opportunities. 2nd Edition, John Wiley & Sons, Ltd, England, 2008.

[2] M. Weiser. The Computer for the Twenty-First Century. Scientific American, Vol. 265, No. 3, 94-104 (1991).

[3] M. Pantic, A. Pentland, A. Nijholt, and T.S. Huang. Human Computing and Machine Understanding of Human Behaviour: A Survey. In: Human Computing. Lecture Notes in Artificial Intelligence 4451, Springer-Verlag Berlin, pp. 47–71 (2007).

[4] A. Nijholt. Where computers disappear, virtual humans appear. Computers & Graphics 28: 467–476 (2004).

[5] A. Nijholt. Google Home: Experience, Support and Re-experience of Social Home Activities. *Information Sciences*. Vol. 178, No. 3, Elsevier, Amsterdam, pp. 612-630 (2008).

[6] A. Nijholt, T. Rist & K. Tuinenbreijer. Lost in ambient intelligence. In: Proceedings ACM Conference on Computer Human Interaction (CHI 2004), ACM, New York, pp. 1725-1726 (2004).

[7] C. Castelfranchi. Artificial Liars: Why Computers Will (Necessarily) Deceive Us and Each Other. Ethics and Inf. Tech. Vol. 2, No. 2, pp. 113-119 (2000).

[8] H. Sacks, E.A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. Language, 50(4), 696-735 (1974).

[9] D. Heylen, A. Nijholt, and M. Poel. Generating Nonverbal Signals for a Sensitive Artificial Listener. In: Verbal and Nonverbal Communication Behaviours. Lecture Notes in Computer Science 4775, Springer-Verlag Berlin Heidelberg, pp. 264-274 (2007).

[10] D. Reidsma, H. van Welbergen, R. Poppe, P. Bos, & A. Nijholt. Towards Bi-directional Dancing Interaction. Proceedings 5th International Conference on Entertainment Computing (ICEC 2006), Cambridge, UK, R. Harper, M. Rauterberg, M. Combetto (Eds.), Lecture Notes in Computer Science 4161, Springer-Verlag, Berlin, Heidelberg, pp. 1-12 (2006).

[11] P. Bos, D. Reidsma, Z. Ruttkay & A. Nijholt. Interacting with a Virtual Conductor. Proceedings 5th International Conference on Entertainment Computing (ICEC 2006), Cambridge, UK, R. Harper, M. Rauterberg, M. Combetto (Eds.), Lecture Notes in Computer Science 4161, Springer-Verlag, Berlin, Heidelberg, pp. 25-30 (2006).

[12] Z.M. Ruttkay & H. van Welbergen. On the timing of gestures of a Virtual Physiotherapist. 3rd Central European Multimedia and Virtual Reality Conference, C.S. Lanyi (eds), Pannonian University Press, Hungary, pp. 219-224 (2006).

[13] D. Heylen, A. Nijholt & R. op den Akker. Affect in Tutoring Dialogues. Journal of Applied Artificial Intelligence (special issue on Educational Agents - Beyond Virtual Tutors), Vol. 19, No. 3-4, Taylor & Francis, pp. 287-311 (2005).

[14] A. Nijholt et al. Mutually Coordinated Anticipatory Multimodal Interaction. In: Nonverbal Features of Human-Human and Human-Machine Interaction. Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, to appear (2008).

[15] R. Rienks, A. Nijholt, D. Reidsma. Meetings and Meeting Support in Ambient Intelligence. Chapter 18 in: Ambient Intelligence, Wireless Networking, Ubiquitous Computing. Th. A. Vasilakos & W. Pedrycz (eds.), Artech House, Norwood, MA, USA, pp. 359-378 (2006).

[16] W. Ju & L. Leifer. The Design of Implicit Interactions: Making Interactive Objects Less Obnoxious. Design Issues: Special Issue on Design Research in Interaction Design. In Press.

[17] S. Babu, C. Zanbaka, J. Jackson, T-O Chung, B. Lok, M.C. Shin, & L.F. Hodges. Virtual Human Physiotherapist Framework for Personalized Training and Rehabilitation, *Graphics Interface 2005*, Victoria, British Columbia, Canada (2005).

[18] S. Mokka, A. Väätänen, J. Heinilä, & P. Välkkynen. Fitness computer game with a bodily user interface. Proceedings of the Second international Conference on Entertainment Computing, Pittsburgh, ACM International Conference Proceeding Series, vol. 38. Carnegie Mellon University, Pittsburgh, PA, pp. 1-3 (2003).

[19] P.T. Chua et al. Training for Physical Tasks in Virtual Environments: Tai Chi. Proceedings of the IEEE Virtual Reality 2003, IEEE Computer Society, Washington, DC, pp. 87-94 (2003).

[20] J. Höysniemi, A. Aula, P. Auvinen, J. Hännikäinen, & P. Hämäläinen. Shadow boxer: a physically interactive fitness game. Proceedings of the Third Nordic Conference on Human-Computer interaction. NordiCHI '04, vol. 82. ACM Press, New York, NY, pp. 389-392 (2004).

[21] P. Hämäläinen et al. Martial Arts in Artificial Reality, Proceedings ACM Conference on Human Factors in Computing Systems (CHI'2005), ACM Press (2005).

[22] F. Müller & S. Agamanolis. Sports over a distance. Comput. Entertain. Vol. 3, No. 3, pp. 4-4, 2005.

[23] B. Meyers et al. Dance your work away: exploring step user interfaces. In CHI '06 Extended Abstracts on Human Factors in Computing Systems, Montréal, CHI '06. ACM Press, New York, NY, pp. 387-392 (2006).

[24] N. Ambady and R. Rosenthal. Thin slices of expressive behavior as predictors of interpersonal consequences: A meta-analysis. Psychological Bulletin, 111, 2, pp. 256-274 (1992).

[25] P. Borkenau, N. Mauer, R. Riemann, F.M. Spinath, and A. Angleitner. Thin Slices of Behavior as Cues of Personality and Intelligence. Journal of Personality and Social Psychology, Vol. 86, No. 4, pp. 599–614 (2004).

[26] Bechinie Bechinie, M., & Grammer, K. Charisma. Cam: A prototype of an intelligent digital sensory organ for virtual humans. Proceedings *IVA 2003*, LNAI 2792, Springer, pp. 212-216 (2003).

[27] R.W. Picard, E. Vyzas, & J. Healey. Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23 (2001), No. 10, pp. 1175-1191 (2001).

[28] K. Gilleade, A. Dix, J. Allanson. Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me. Proceedings of DIGRA'2005, Vancouver, Canada (2005).

[29] D.S. Tan. Brain-Computer Interfaces: applying our minds to human-computer interaction. Informal proceedings "What is the Next Generation of Human-Computer Interaction?" Workshop at CHI 2006, Montreal (2006).

Private Information and Inference about Inference: Experiments with Human Subjects and Computer Agents

Sobei H. Oda and **Gen Masumoto** and **Hiroyasu Yoneda**[1]

Abstract. This paper describes how players infer other players' inference in the game of Schredelseker [7]. Schredelseker did a series of laboratory experiments and simulations of his zero-sum game played by players who are informed to different degrees, reporting that modestly informed players often suffered from greater losses than no-informed ones did. This paper formulates each player's profit rigorously to examine why such paradoxical income distribution was observed. In addition, mathematical analysis and experiments with human subjects and/or computer agents are given to illustrate how players' inter-inference about other players' inference affects the distribution of income among them.

1 Introduction

Let us assume that speculators trade future commodities among them. Since they repeat buying and selling commodities among them without producing or consuming anything, speculative traders play a zero-sum game: (A) the sum of all traders' profit is zero. Nevertheless those who have private information which may affect future prices can make use of it to determine whether they buy or sell future commodities. It seems certain that (B) more informed traders have more chance to make more profit than less informed traders have. The least informed speculators, who have no private information, would have no other way but to buy or sell randomly. However, does it give them equal chance to make money (to buy a commodity whose price will increase or to sell a commodity whose price will decrease) and to lose money (to sell a commodity whose price will increase or to buy a commodity whose price will decrease)? If so, (C) no-informed traders can expect zero profit, which is inconsistent with (A) and (B).

Schredelseker [7] formulated a simple model to describe speculative trading among traders with different private information. He did a series of its simulations and experiments to find that (B) is not always the case: modestly informed players quite often suffered from greater losses than no-informed players did; see also [2]. This paper examines why such paradoxical distribution of profit were observed. The point is that modestly informed players often use their private information without considering other players' behaviour properly. If you are informed enough to predict future prices perfectly or have no information about future prices, you need not or cannot consider your rivals' strategies: you have only to buy those commodities whose prices will rise and sell those commodities whose prices will fall or you have nothing to think about. However, if you are modestly informed, you must take other players' strategies into account to maximise their advantage over less informed rivals while minimising their disadvantage against more informed ones, which calculation could be too demanding for you to solve.

This paper is an extended abstract of our recent studies [6], which explains details of mathematical analysis of Schredelseker's model and Yoneda, Masumoto and Oda [10], which explain the results of its experiments with computer agents as well as with computer agents and human subjects. We hope that new results and findings will be included in the final paper.

In this paper we shall present some mathematical formulations of players' inference about other players' inference as well as new computer simulations to illustrate what dynamics players' inter-inference will generate. In Section 2 we shall formulate Schredelseker's model rigorously with small modifications. In Section 3 we shall examine the model to show the mechanism that distorts (B). In Section 4 we shall formulate a series of players' inference about other players' inference in the model. We shall define players' inference about other players' inference in the first subsection; analyse it mathematically for a small number of players in the second subsection; introduce additional assumptions to define their inference about other players' inference about other players' inference, ... in the third subsection; show some results of simulations to illustrate what dynamics the series of inference about inference could generate. In Section 5 we shall mention results of experiments with computer agents and human subjects.

2 Schredelseker's model

The model of Schredelseker (2007) is summarised in the following.

1. The price of a commodity in the spot market of Day Two (tomorrow) Q is determined to be the sum of $2N$ stochastic variables $X_1, X_2, ...,$ and X_{2N} which are zero or unity independently with equal probability:
$$Q = \sum_{i=1}^{2N} x_i. \tag{1}$$
Here x_i stands for the realised value of X_i, which equals 0 or 1. The accumulate distribution function of Q is defined as
$$\text{Prob}(Q \le q) = \frac{1}{2^{2N}} \sum_{i=0}^{q} \frac{2N!}{(2N-i)!i!}. \tag{2}$$
2. There are $2N+1$ speculative traders (players) who buy or sell the commodity among them in the future market of Day One (today). Although the equilibrium price of the tomorrow's spot market Q is not realised before tomorrow, $x_0, x_1, ...,$ and x_{2N} are all determined before the today's future market is open and the first k of them are known to Player k before she determines R_k: Player 0

[1] Kyoto Sangyo University, Japan, email: oda@cc.kyoto-su.ac.jp

knows none of $x_1, x_2, ...,$ and x_{2N}; Player 1 knows $X_1 = x_1$; Player 2 knows $X_1 = x_1$ and $X_2 = x_2$; ...; Player $2N$ knows $X_1 = x_1, X_2 = x_2, ...,$ and $X_2 = x_{2N}$. In short, Player k is unconditionally better informed than Player $k-1$ in the meaning that what the latter knows is all known to the former.[2]

3. All players declares their reservation prices (ask/bid prices): R_0, R_1, ..., and R_{2N+1} where R_k stands for the reservation price of Player k. Once she announces her reservation price to be R_k, Player k must buy a unit of the commodity if R_k is higher than the equilibrium price of the today's future market P while she must sell a unit if $R_k < P$ (she may be asked to buy a unit or to sell a unit or to trade nothing if $R_k = P$).
4. The equilibrium price of the today's future market P is determined to be the median of all the $2N+1$ players' reservation prices:
$$P = R_{i_N} \tag{3}$$
where $(i_0, i_1, \cdots, i_{2N})$ is a permutation of $(0, 1, ..., 2N)$ that satisfies
$$R_{i_0} \le R_{i_1} \le \cdots \le R_{i_{N-1}} \le R_{i_N} \le R_{i_{N+1}} \le \cdots \le R_{i_{2N-1}} \le R_{i_{2N}} . \tag{4}$$
Players i_0, i_2, ..., and i_{N-1} sell to Players i_{N+1}, i_{N+2}, ..., and i_{2N} (a unit from a seller to a buyer at Price P) while Player i_N does not take part in trade. The person who does not trade may not be determined uniquely, because there may exist two or more permutations of 0,1, ..., and $2N$ that satisfy (4). In such cases one of them is chosen randomly to chose the player who does not trade, but P is always determined uniquely by (3):[3]
$$R_{i_N} = \mu(R_0, R_1, \cdots, R_{2N}). \tag{5}$$
5. The $2N$ players who buy or sell in the today's future market trade reversely in the tomorrow's spot market at price Q to offset their speculative trades. As a result, the profit of Player k is determined as
$$\Pi_k(P, Q, R_k)$$
$$\begin{cases} = P - Q & \text{if } R_k < P \text{ or } (R_k = P \text{ and } k \neq i_N) \\ = 0 & \text{if } k = i_N \\ = Q - P & \text{if } P < R_k \text{ or } (P = R_k \text{ and } k \neq i_N) \end{cases}, \tag{6}$$
which implies that the $2N+1$ players play a zero-sum game:
$$\sum_{i=0}^{2N} \Pi_i(P, Q, R_i) = 0. \tag{7}$$

We assume that what is mentioned above is known to all players as common knowledge.

3 Players' Expectation of Profit

What Player k can and must determine is the value of her reservation price R_k; her profit is determined according to all player's reservation prices including her own one. In other words, her strategy is a mapping from her private information set $\{X_1 = x_2, X_2 = x_2, \cdots X_k = x_k\}$ to her action R_k. What is the strategy that maximises the expected value of her profit?

Let us assume Player k expects that P is not greater than p with probability
$$\text{Prob}_k(P \le p) = \int_0^p f_k(p; \boldsymbol{x}_k, R_k) dp \tag{8}$$
and that $Q = q$ with probability
$$\text{Prob}_k(Q = q) = g_k(q; \boldsymbol{x}_k) \tag{9}$$
where
$$\boldsymbol{x}_k = (x_1, x_2, \cdots, x_k) . \tag{10}$$
Here the following is assumed: Player k correctly regards P as a continuous variable and Q as a discrete variable; she may utilise her private information $\boldsymbol{x}_k$ to estimate $\text{Prob}_k(P \le p)$ and $\text{Prob}_k(Q = q)$; she may take account of the effect of her own decision R_k on $\text{Prob}_k(P \le p)$.

Player k can calculate the following conditional probability from her private information.
$$\text{Prob}\left(Q = q \,\middle|\, \begin{matrix} X_1 = x_1 \\ X_2 = x_2 \\ \vdots \\ X_k = x_k \end{matrix} \right)$$
$$\begin{cases} = 0 & \text{for } \; q < \sum_{i=1}^{k} x_i \\ = \dfrac{\left\{ \dfrac{(2N-k)!}{(2N-k-q)!q!} \right\}}{2^{2N-k}} & \text{for } \; \sum_{i=1}^{k} x_i \le q \le 2N - k + \sum_{i=1}^{k} x_i \\ = 0 & \text{for } \; 2N - k + \sum_{i=1}^{k} x_i < q \end{cases} . \tag{11}$$

We assume that she adopts the objective probability (11) as her subjective probability about Q:
$$g_k(Q; \boldsymbol{x}_k)$$
$$\begin{cases} = 0 & \text{for } \; Q < Q_k^{\min}(\boldsymbol{x}_k) \\ = \dfrac{1}{2^{2N-k}} \dfrac{(2N-k)!}{(2N-k-Q)!Q!} & \text{for } \; Q_k^{\min}(\boldsymbol{x}_k) \le Q \le Q_k^{\max}(\boldsymbol{x}_k) \\ = 0 & \text{for } \; Q_k^{\max}(\boldsymbol{x}_k) < Q \end{cases} \tag{12}$$
where[4][5]
$$Q_k^{\min}(\boldsymbol{x}_k) = \sum_{i=1}^{k} x_i \quad \text{and} \quad Q_k^{\max}(\boldsymbol{x}_k) = 2N - k + \sum_{i=1}^{k} x_i. \tag{13}$$

[2] In the text we often designate a typical player as Player k. For convenience we assume Player k is female while her competitors are all male.

[3] Though the number of players is an even number in the original model, we assume it is an odd number to determine P uniquely. In addition to the tie-break rule mentioned in the text, there can be several tie-break rules that determine who trade and who do not if two or more traders have P_{i_N} as their reservation prices. The analysis of the text does not change essentially whichever rule is assumed.

[4] It is not essential for our formulation that Player k regards the conditional probability as her subjective probability. If the subjective probability $g_k(Q; \boldsymbol{x}_k)$ differs the objective probability given by right-hand side of (12), the analysis in the text holds true without modification.

[5] Since $\boldsymbol{x}_0$ is not defined, we must add $Q_0^{\min} = 0$ and $Q_0^{\max} = N$ to (13) to define $Q_k^{\min}$ and $Q_k^{\max}$ for all k $(0 \le k \le 2N)$. To keep explanation simple, however, we do not mention such natural stretching to the cases where $k = 0$ in the text; such irrational expressions as $Q_0^{\min}(\boldsymbol{x}_0)$ and $Q_0^{\max}(\boldsymbol{x}_0)$ should be read as $Q_0^{\min}$ and $Q_0^{\max}$.

Let $I_k(\boldsymbol{x}_k)$ be the closed interval between $Q_k^{\min}(\boldsymbol{x}_k)$ and $Q_k^{\max}(\boldsymbol{x}_k)$:

$$I_k(\boldsymbol{x}_k) = \left\{ Q \,\middle|\, Q_k^{\min}(\boldsymbol{x}_k) \leq Q \leq Q_k^{\max}(\boldsymbol{x}_k) \right\}. \tag{14}$$

Player k, who knows (6) and (12), can see that choosing R_k outside $I_k(\boldsymbol{x}_k)$ is weakly dominated by choosing it at the either end of the interval:

$$\begin{gathered} R_k < Q_k^{\min}(\boldsymbol{x}_k) \\ \Downarrow \\ \Pi_k(P,Q,R_k) \leq \Pi_k(P,Q,Q_k^{\min}(\boldsymbol{x}_k)) \quad \text{for all } P \end{gathered} \tag{15}$$

and

$$\begin{gathered} Q_k^{\max}(\boldsymbol{x}_k) < R_k \\ \Downarrow \\ \Pi_k(P,Q,R_k) \leq \Pi_k(P,Q,Q_k^{\max}(\boldsymbol{x}_k)) \text{for all } P. \end{gathered} \tag{16}$$

We assume that Player k chooses R_k in I_k not to choose a weakly dominated strategy:

$$Q_k^{\min}(\boldsymbol{x}_k) \leq R_k \leq Q_k^{\max}(\boldsymbol{x}_k). \tag{17}$$

In addition we assume that Player k considers that none of her rivals choose a dominated strategy either:

$$Q_h^{\min}(\boldsymbol{x}_h) \leq R_h \leq Q_h^{\max}(\boldsymbol{x}_h) \quad \text{for all } h\ (h \neq k), \tag{18}$$

which implies that

$$0 \leq P \leq P^{\max} = 2N. \tag{19}$$

Incidentally it should be noted that (17) implies that

$$0 < \Pi_k(P,Q,R_k) = |P-Q| \quad \text{if } R_k \in I_k(\boldsymbol{x}_k) \text{ and } P \notin I_k(\boldsymbol{x}_k), \tag{20}$$

which can be another reason why we assume (refequation: interval of Rh) and is an expression of the advantage of better informed players because

$$\begin{aligned} \{Q\} &= I_{2N}(\boldsymbol{x}_{2N}) \subset I_{2N-1}(\boldsymbol{x}_{2N-1}) \subset \cdots \subset I_1(\boldsymbol{x}_1) \subset I_0 \\ &= \{Q \,|\, 0 \leq Q \leq 2N\}. \end{aligned} \tag{21}$$

In particular, Player $2N$, who is perfectly informed, has no chance to suffer from losses as long as he keeps $R_{2N} = Q$.

From (8), (12), (17) and (19), the values of P and Q that Player k who has private information $\boldsymbol{x}_k$ expects subjectively are

$$P_k^\star(\boldsymbol{x}_k, R_k) = \int_0^{P^{\max}} f_k(P;\boldsymbol{x}_k,R_k) P dP \tag{22}$$

and

$$Q_k^\star(\boldsymbol{x}_k) = \sum_{Q_k^{\min}}^{Q_k^{\max}} g_k(Q;\boldsymbol{x}_k) Q = \frac{2N-k}{2} + \sum_{i=1}^{k} x_i \tag{23}$$

respectively. Hence the profit that Player k expects subjectively is expressed as

$$\begin{aligned} \Pi_k^\star(\boldsymbol{x}_k, R_k) \;=\; & \int_0^{P^{\max}} f_k(P;\boldsymbol{x}_k,R_k) \\ & \sum_{Q=Q_k^{\min}(\boldsymbol{x}_k)}^{Q_k^{\max}(\boldsymbol{x}_k)} \{g_k(Q;\boldsymbol{x}_k)\Pi_k(P,Q,R_k)\}\, dP, \end{aligned} \tag{24}$$

from which we can obtain

$$\begin{aligned} & \frac{\partial \Pi_k^*(\boldsymbol{x}_k, R_k)}{\partial R_k} \\ & = 2\{f_k(R_k;\boldsymbol{x}_k,R_k) + \phi_k(R_k;\boldsymbol{x}_k,R_k)\}\{Q_k^\star(\boldsymbol{x}_k) - R_k\} \\ & - 2F_k(R_k;\boldsymbol{x}_k,R_k) + 2\frac{\partial}{\partial R_k}\mathcal{F}_k(R_k;\boldsymbol{x}_k,R_k) \\ & - \frac{\partial}{\partial R_k}\mathcal{F}_k(P^{\max};\boldsymbol{x}_k,R_k). \end{aligned} \tag{25}$$

Here it is assumed that the following functions exist.

$$F_k(P;\boldsymbol{x}_k,R_k) = \int_0^P f_k(p;\boldsymbol{x}_k,R_k)dp \tag{26}$$

$$\mathcal{F}_k(P;\boldsymbol{x}_k,R_k) = \int_0^P F_k(p;\boldsymbol{x}_k,R_k)dp \tag{27}$$

$$\phi_k(P;\boldsymbol{x}_k,R_k) = \frac{\partial F_k(P;\boldsymbol{x}_k,R_k)}{\partial R_k} \tag{28}$$

If she considers that her choice of R_k does not affect P, Player k will choose $Q_k^\star(\boldsymbol{x}_k)$ because, in addition that it is in $I_k(\boldsymbol{x}_k)$, (25) implies that

$$\begin{gathered} f_k(P;\boldsymbol{x}_k,R_k) = f_k(P;\boldsymbol{x}_k) \\ \Downarrow \\ 0 \lesseqgtr \frac{\partial \Pi_k^*(\boldsymbol{x}_k,R_k)}{\partial R_k} \quad \Leftrightarrow \quad R_k \lesseqgtr Q_k^\star(\boldsymbol{x}_k). \end{gathered} \tag{29}$$

However, since R_k is one of the reservation prices that determines P as their median, what value Player k chooses as R_k can change the value of P. If she takes it into account, Player k can see that $R_k = Q_k^\star$ does not maximise her profit, though (24) may be so complicated that she could not discover the value of R_k that maximises the expected value of her profit.

4 Taking Competitors' Strategies into Account

4.1 Level One and Two Strategies

Let us define Player k's Level One Strategy $\mathbf{L1}_k$ as choosing her expectation of Q as her reservation price, and her Level Two Strategy $\mathbf{L2}_k$ as choosing such a reservation price that maximises the expectation of her profit on the supposition that her competitors all follow their level two strategies:

$$\mathbf{L1}_k : R_k = Q_k^\star(\boldsymbol{x}_k) \tag{30}$$

$$\mathbf{L2}_k : R_k \in I_k^{(2)}(\boldsymbol{x}_k) \tag{31}$$

where

$$I_k^{(2)}(\boldsymbol{x}_j) = \left\{ R_j \,\middle|\, \begin{gathered} R_j = Q_j^\star(\boldsymbol{x}_j) \quad \text{for all } j\ (j \neq k) \\ \Downarrow \\ \Pi_k^\star(\boldsymbol{x}_k, r_k) \leq \Pi_k^\star(\boldsymbol{x}_k, R_k) \quad \text{for all } r_k \end{gathered} \right\}. \tag{32}$$

Player k can obtain $\mathbf{L2}_k$ in the following way.

1. Since she can see $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., and $\boldsymbol{x}_{k-1}$ from her private information $\boldsymbol{x}_k$, Player k can readily see $Q_0^\star$, $Q_1^\star(\boldsymbol{x}_1)$, ..., and $Q_{k-1}^\star(\boldsymbol{x}_{k-1})$.
2. Player k calculates the reservation prices of Players $k+1, k+2$, ..., and $2N$ for every possible combinations of the realised values of X_{k+1}, X_{k+2}, ..., and X_{2N}:

$$Q_h^\star(\boldsymbol{z}_k^h) = \frac{2N-h}{2} + \sum_{i=1}^{k} x_i + \sum_{i=k+1}^{h} y_i \tag{33}$$

where y_i ($k+1 \leq i \leq 2N$) stands for the value of X_i which is realised but unknown to Player k and

$$\boldsymbol{z}_k^h = (x_1, x_2, \cdots, x_k, y_{k+1}, \cdots, y_h). \tag{34}$$

3. Player k calculates P for every $\boldsymbol{z}_k^{2N}$ and R_k :

$$\begin{aligned} P_k^\star(R_k; \boldsymbol{z}_k^{2N}) &= \mu\big(N, Q_1^\star(\boldsymbol{x}_1), \cdots, Q_1^\star(\boldsymbol{x}_{k-1}), R_k, \\ &\qquad Q_{k+1}^\star(\boldsymbol{z}_k^{k+1}), \cdots, Q_{2N}^\star(\boldsymbol{z}_{2N}^h)\big) . \end{aligned} \tag{35}$$

4. Player k calculates the expectation of her profit for every $\boldsymbol{z}_k^{2N}$ and R_k:

$$\Pi_k^\star(R_k, \boldsymbol{z}_k^{2N}) = \Pi_k(P_k^\star(R_k; \boldsymbol{z}_k^{2N}), Q_h^\star(\boldsymbol{z}_k^h), R_k). \tag{36}$$

5. Since each $\boldsymbol{z}_k^{2N}$ is realised with equal probability, she can calculate her profit that she can expect when her private information for every $\boldsymbol{x}_k$ and R_k:

$$\Pi_k^\star(R_k; \boldsymbol{x}_k) = \frac{1}{2N-k} \sum_{y_{k+1}=0}^{1} \sum_{y_{k+2}=0}^{1} \cdots \sum_{y_{2N}=0}^{1} \Pi_k^\star(R_k, \boldsymbol{z}_k^{2N}). \tag{37}$$

6. Player k chooses such R_k that maximises $\Pi_k^\star(R_k; \boldsymbol{x}_k)$ as her second level strategy.

4.2 Example

Let us examine the case where there are five players ($N = 2$). We shall construct each player's level two strategy in the following paragraphs, summarising its implications at the end of this subsection.

As an example we outline how to obtain Player One's level two strategy. Player One can see the value of x_1. If he knows $x_1 = 0$, on the supposition that Players Zero, Two, Three and Four all follow their level one strategies, Player One can see that one of the following eight cases is realised with equal probability.

$$\begin{pmatrix} x_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \Rightarrow \quad \left\{ \begin{array}{rcl} R_0 &=& 2 \\ R_2 &=& 1 \\ R_3 &=& 0.5 \\ Q = R_4 &=& 0 \end{array} \right.$$

$$\Rightarrow \quad \Pi_1 \begin{cases} = 0.5 & \text{for } R_1 < 0.5 \\ = 0.25 & \text{for } R_1 = 0.5 \\ = 0 & \text{for } 0.5 < R_1 < 1 \\ = -0.5 & \text{for } R_1 = 1 \\ = -1 & \text{for } 1 < R_1 \end{cases} \tag{38}$$

$$\begin{pmatrix} x_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \left\{ \begin{array}{rcl} R_0 &=& 2 \\ R_2 &=& 1 \\ R_3 &=& 0.5 \\ Q = R_4 &=& 1 \end{array} \right. \tag{39}$$

$$\Rightarrow \quad \Pi_1 = 0 \quad \text{for all } R_1$$

$$\vdots$$

$$\begin{pmatrix} x_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \left\{ \begin{array}{rcl} R_0 &=& 2 \\ R_2 &=& 2 \\ R_3 &=& 2.5 \\ Q = R_4 &=& 3 \end{array} \right.$$

$$\Rightarrow \quad \Pi_1 \begin{cases} = -1 & \text{for } R_1 < 2 \\ = -\frac{2}{3} & \text{for } R_1 = 2 \\ = 0 & \text{for } 2 < R_1 < 2.5 \\ = 0.25 & \text{for } R_1 = 2.5 \\ = 0.5 & \text{for } 2.5 < R_1 \end{cases} \tag{40}$$

from which Player One can express the expectation of his profit as a function of R_1:

$$\begin{array}{c} x_1 = 0 \\ \Downarrow \end{array}$$

$$\Pi_1^\star \begin{cases} = \frac{0.5+0+\cdots-1}{8} = -0.0625 & \text{for } R_1 < 0.5 \\ = \frac{0.25+0+\cdots-1}{8} = -0.09375 & \text{for } R_1 = 0.5 \\ = \frac{0+0+\cdots-1}{8} = -0.125 & \text{for } 0.5 < R_1 < 1 \\ = \frac{-0.5+0+\cdots-1}{8} = -0.1875 & \text{for } R_1 = 1 \\ = \frac{-1+0+\cdots-1}{8} = -0.25 & \text{for } 1 < R_1 < 1.5 \\ = \frac{-1+0+\cdots-1}{8} = -0.28125 & \text{for } R_1 = 1.5 \\ = \frac{-1+0+\cdots-1}{8} = -0.3125 & \text{for } 1.5 < R_1 < 2 \\ = \frac{-1+0+\cdots-\frac{2}{3}}{8} = -0.3541666\cdots & \text{for } R_1 = 2 \\ = \frac{-1+0+\cdots+0}{8} = -0.3125 & \text{for } 2 < R_1 < 2.5 \\ = \frac{-1+0+\cdots+0.25}{8} = -0.28125 & \text{for } R_1 = 2.5 \\ = \frac{-1+0+\cdots+0.5}{8} = -0.25 & \text{for } 2.5 < R_1 \end{cases} . \tag{41}$$

He can also see his profit that is expected if $x_1 = 1$, which is symmetric with (41):

$$x_1 = 1 \quad \Rightarrow \quad \Pi_1^\star \begin{cases} = -0.25 & \text{for } R_1 < 1.5 \\ = -0.28125 & \text{for } R_1 = 1.5 \\ = -0.3125 & \text{for } 1.5 < R_1 < 2 \\ = -0.3541666\cdots & \text{for } R_1 = 2 \\ = -0.3125 & \text{for } 2 < R_1 < 2.5 \\ = -0.28125 & \text{for } R_1 = 2.5 \\ = -0.25 & \text{for } 2.5 < R_1 < 3 \\ = -0.1875 & \text{for } R_1 = 3 \\ = -0.125 & \text{for } 3 < R_1 < 3.5 \\ = -0.09375 & \text{for } R_1 = 3.5 \\ = -0.0625 & \text{for } 3.5 < R_1 \end{cases} . \tag{42}$$

From (41) and (42) Player One obtains his level two strategy:

$$I_1^{(2)}(\boldsymbol{x}_1) \begin{cases} = \{R_1 \,|(0 \leq) R_1 < 0.5\} & \text{if } \boldsymbol{x}_1 = (0) \\ = \{R_1 \,|3.5 < R_1 (\leq 4)\} & \text{if } \boldsymbol{x}_1 = (1) \end{cases} \tag{43}$$

where inequalities in parentheses are imposed by (17).

Level two strategies for Players Zero, Two, Three and Four can be obtained similarly:

$$I_0^{(2)} = \{R_0 \,|(0 \leq) R_0 < 0.5\} \cup \{R_0 \,|3.5 < R_0 (\leq 4)\} \tag{44}$$

$$I_2^{(2)}(\boldsymbol{x}_2) \begin{cases} = \{R_2 \,|(0 \leq) R_2 < 0.5\} & \text{if } \boldsymbol{x}_2 = (0,0) \\ = \{R_2 \,|2.5 < R_2 (\leq 3)\} & \text{if } \boldsymbol{x}_2 = (0,1) \\ = \{R_2 \,|(1 \leq) R_2 < 1.5\} & \text{if } \boldsymbol{x}_2 = (1,0) \\ = \{R_2 \,|3.5 < R_2 (\leq 4)\} & \text{if } \boldsymbol{x}_2 = (1,1) \end{cases} \tag{45}$$

$$I_3^{(2)}(\boldsymbol{x}_3)\begin{cases} = \{R_3\,|(0\leq)R_3 < 1\} & \text{if } \boldsymbol{x}_3 = (0,0,0)\\ = \{R_3\,|(1\leq)R_3(\leq 2)\} & \text{if } \boldsymbol{x}_3 = (0,0,1)\\ = \{R_3\,|(1\leq)R_3 < 1.5\} & \text{if } \boldsymbol{x}_3 = (0,1,0)\\ = \{R_3\,|(2\leq)R_3(\leq 3)\} & \text{if } \boldsymbol{x}_3 = (0,1,1)\\ = \{R_3\,|(1\leq)R_3 < 2\} & \text{if } \boldsymbol{x}_3 = (1,0,0)\\ = \{R_3\,|2.5 < R_3(\leq 3)\} & \text{if } \boldsymbol{x}_3 = (1,0,1)\\ = \{R_3\,|(2\leq)R_3(\leq 3)\} & \text{if } \boldsymbol{x}_3 = (1,1,0)\\ = \{R_3\,|3 < R_3(\leq 4)\} & \text{if } \boldsymbol{x}_3 = (1,1,1)\end{cases} \tag{46}$$

$$I_4^{(2)}(\boldsymbol{x}_4) = \{R_4\,|R_4 = x_1+x_2+x_3+x_4\} \quad \text{for all } \boldsymbol{x}_4. \tag{47}$$

Here inequalities in parentheses are imposed by (17).

Having seen all players' level two strategies, we could surmise the following as general properties which would hold true for $3\leq N$.[6]

(a). Although a player may follow his/her level one strategy to maximise his/her profit when his/her competitors determine their reservation prices thoughtlessly, the performance of the level one strategy is rather poor — in fact it is often the worst in the meaning that it minimises the expectation of his/her profit — when all competitors follow their level one strategies.

(b). Player k's level two strategy is determined not uniquely as a certain value of R_k but as an interval or a union of intervals of R_k. Even if (17) is imposed, no interval but $I_{2N}^{(2)}(\boldsymbol{x}_{2N})$ degenerates to a point.

(c). Buying at any price ($R_k = \infty$) or selling at any price ($R_k = 0$) can be Player k's level two strategy, namely $R_k = Q_k^{\min}(\boldsymbol{x}_k)$ or $R_k = Q_k^{\max}(\boldsymbol{x}_k)$ under the condition of (17), can be her level two strategy. In our example ($N = 2$), as we have seen in this subsection, either $R_k = Q_k^{\min}(\boldsymbol{x}_k)$ or $R_k = Q_k^{\max}(\boldsymbol{x}_k)$ can be the Player k's level two strategy except for $k = 0$ and $k = 2N$. Yet it is not always the case for larger N. For example, both $R_k = Q_k^{\min}(\boldsymbol{x}_k)$ and $R_k = Q_k^{\max}(\boldsymbol{x}_k)$ can be the level two strategy of Player 3 if $N = 3$ and $\boldsymbol{x}_3 = (0,0,1)$.

(d). Unlike level one strategies, level two strategies depends on the permutation of x_1, x_2, ..., and x_k. As an example we saw that $Q_2^\star((1,0)) = Q_2^\star((0,1)) = 2$ while $I_2^{(2)}((1,0))\cap I_2^{(2)}((0,1)) = \emptyset$ for $N = 2$.

(e). Player Zero, who has no private information may be able to reduce his disadvantage if he realises how other players use their private information to which he cannot access. In our example, if he is aware that more informed players all follow their level one strategy, Player Zero can change his strategy from $\mathbf{L1}_0$ to $\mathbf{L2}_0$ to reduce the expectation of his loss from 0.25 to 0.125.

4.3 Additional Rules for Simulations

Let $R_k^l(\boldsymbol{x}_k)$ be the value of R_k that Player k chooses as her Level l strategy. It would be straightforward to define $R_k^{(3)}(\boldsymbol{x}_k)$ if $R_k^{(2)}(\boldsymbol{x}_k)$ is determined uniquely: $\mathbf{L3}_k(\boldsymbol{x}_k)$ would be defined as her strategy that maximises her expectation of profit on the supposition that her competitors all follow their level two strategies. However, (b) implies that $R_k^{(2)}(\boldsymbol{x}_k)$ is not determined uniquely even if (17) is imposed.

In the circumstances we assume the following in our simulations to define $R_k^{(2)}(\boldsymbol{x}_k)$ uniquely.

[6] what is mentioned in the text is checked for larger numbers of N by numerical calculations, though it does not assure that it holds for all N.

Rule 1$^{(2)}$ Define $J_k^{(2)}(\boldsymbol{x}_k)$ as the set of R_k that belongs to $I_k^{(2)}(\boldsymbol{x}_k)$ and a multiple of $\frac{1}{4}$:

$$J_k^{(2)}(\boldsymbol{x}_k) = I_k^{(2)}(\boldsymbol{x}_k) \cap \left\{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \cdots \frac{8N}{4}\right\}. \tag{48}$$

If $J_k^{(2)}(\boldsymbol{x}_k)$ contains only one element, let it be $R_k^{(2)}(\boldsymbol{x}_k)$.

Rule 2$^{(2)}$ If $J_k^{(2)}(\boldsymbol{x}_k)$ contains two or more elements, eliminate such points that are not nearest to $R_k^{(1)}(\boldsymbol{x}_k)$. If there remains only one element as a result, let it be $R_k^{(2)}(\boldsymbol{x}_k)$.

Rule 3$^{(2)}$ If two R_k remain, choose the smaller one as $R_k^{(2)}(\boldsymbol{x}_k)$.

The rules mentioned above defines $R_k^{(2)}(\boldsymbol{x}_k)$ uniquely. We have only to remember the analysis of the previous subsection to confirm it. Since $Q_j^\star(\boldsymbol{x}_j)\in\{0, \frac{1}{2}, \frac{2}{2}, \cdots, \frac{4N}{2}\}$, Π_k remains constant for all R_k between $\frac{m}{2}$ and $\frac{m+1}{2}$. Hence if Π_k is maximised at a certain value of R_k between $\frac{m}{2}$ and $\frac{m+1}{2}$, it is maximised also at $R_k = \frac{2m+1}{4}$, which assures, together with $I_k^{(2)}(\boldsymbol{x}_k) \neq \emptyset$, that $J_k^{(2)}(\boldsymbol{x}_k) \neq \emptyset$.

Once $R_j^{(2)}(\boldsymbol{x}_j)$ are all determined uniquely, we can define $R_k^{(3)}(\boldsymbol{x}_k)$ uniquely in the following way. First, we can define $I_k^3(\boldsymbol{x}_k)$ for $R_0^{(2)}$, $R_1^{(2)}(\boldsymbol{x}_1)$, ..., and $R_{2N}^{(2)}(\boldsymbol{x}_{2N})$ just as we define $I_k^2(\boldsymbol{x}_k)$ for $R_0^{(1)}$, $R_1^{(1)}(\boldsymbol{x}_1)$, ..., and $R_{2N}^{(1)}(\boldsymbol{x}_{2N})$. Since $R_j^{(2)}(\boldsymbol{x}_j)\in J_k^{(2)}(\boldsymbol{x}_k)$, we can define $J_k^{(3)}(\boldsymbol{x}_k)$ as $I_k^{(3)}(\boldsymbol{x}_k) \cup \{0, \frac{1}{8}, \frac{2}{8}, \cdots, \frac{16N}{8}\}$, which is not empty and thus from which we can choose the smallest value of R_k that are closest to $R_k^{(2)}(\boldsymbol{x}_k)$ as the unique $R_k^{(2)}(\boldsymbol{x}_k)$. We can repeat these procedures to define $R_k^{(4)}(\boldsymbol{x}_k) = \frac{m_{k_4}}{16}$, $R_k^{(5)}(\boldsymbol{x}_k) = \frac{m_{k_5}}{32}$, ...

4.4 Simulations

Let us mention some results of our simulations. To save space, we shall summarise them in five figures with minimal captions. First we shall see how income distribution changes among players as they adopt their higher level strategies; (B) mentioned in Introduction does not hold in Figure 1 ($N = 2$ or the number of players is five) and in Figure 2 ($N = 5$ or the number of players is eleven). Then we shall see (C) mentioned in Introduction does not hold true in Figure 3 ($N = 2$) and Figure 4 ($N = 5$), examine the mechanism that decreases random players' profit less than zero. Last we shall mention in Figure 5 that $\mathbf{L1}_k$ may be the strategy that maximises the expectation of Player k's profit if her competitors all choose their reservation prices randomly.

5 Experiments

We did experiments with computer agents and human subjects at Kyoto Experimental Economics Laboratory (KEEL), Kyoto Sangyo University (KSU) on October 13 and 16, 2004. In total 46 undergraduates of KSU played the game mentioned in the previous section as a unique human player with 100 computer agents. To put it concretely, each subject played the game (a) with 100 level one strategy agents for 100 rounds as Player 100; (b) with 100 stage one strategy agents for 100 rounds as Player 30; (c) with 100 stage strategy for 100 rounds as Player 30. Twenty one subjects played the three sessions in the above-mentioned order, while the other subjects played the last two sessions reversely.

Figures 6(Left) and 6(Right) show how gain and loss were distributed among human subjects and computer agents in Sessions (b) and (c) respectively. There the horizontal and vertical axes are the same as in the previous figures; the data in Figure 6(Left) is aggregated from Session (b) played as the second session and Session (b)

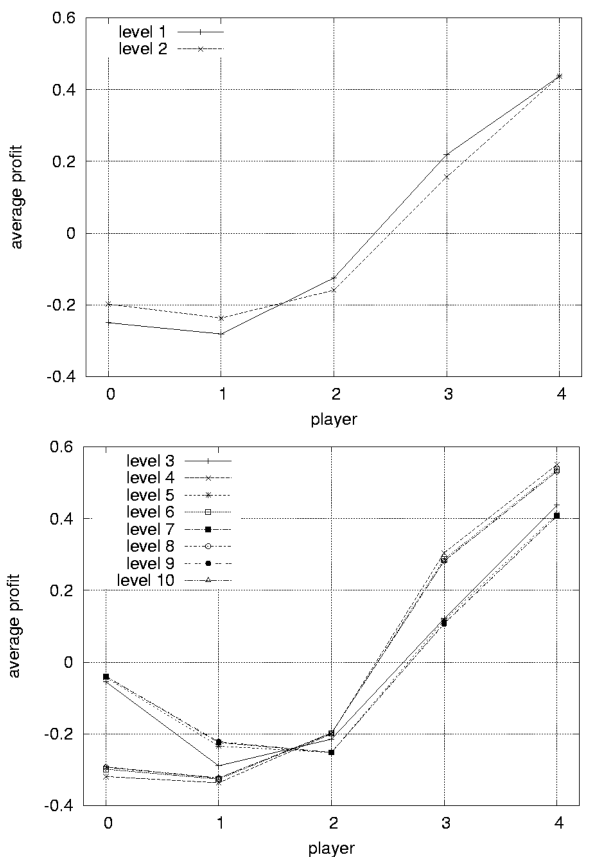


Figure 1. Distribution of profit among players (the number of players is five).

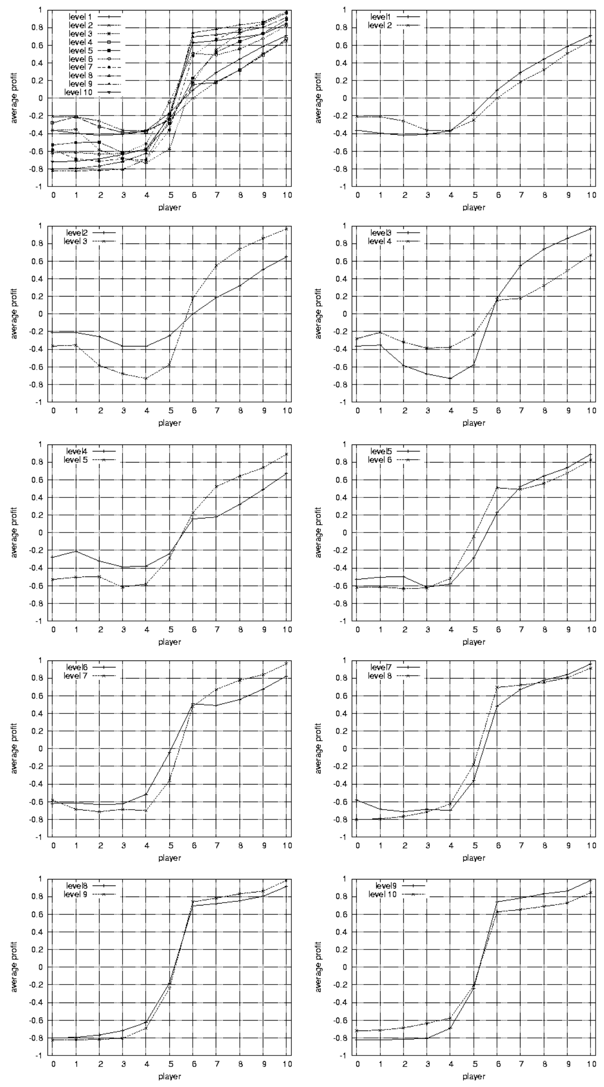


Figure 2. Distribution of profit among players (the number of players is eleven).

as the third session, because the performance of players (subjects and agents) are not different between the two sessions; a similar remark applies to the data in Figure 6(Right). An decrease in loss or increase in gain of Player 30 is visible in Figures 6(Left) and 6(Right), which fact suggests that our subjects changed their strategies according to the strategy of their competitors.

6 Concluding Remarks

As Keynes mentioned in Chapter Six of *General Theory*, "professional investment may be linked to those newspaper competitions in which the competitors have to pick out the six prettiest faces from a hundred photographs, the price being awarded to the competitor whose choice most nearly corresponds to the average preferences of the competitors as a whole." ([3], pp. 156) This comparison to Keynes' beauty contest is often referred to in economic literatures as an illustration of the complexity and instability of a system composed of those subjects who can think and who knows others can think too.

A simplified version of Keynes' beauty contest is the $p-$beauty contest ([4], [5], [1]), where players can infer other players' inference, other players' inference about other players' inference, ... to reach a unique Nash equilibrium. Like the $p-$beauty contest, Schredelseker's model is a zero-sum game where players' inference about inference plays an important role, but it presupposes an hierarchy of private information: Player k is better informed than Player $k-1$, who is better informed than Player $k-2$, ... This presumption seems to be a good abstraction from actual economies consisting not of perfectly informed rational traders and random traders who have no information and no power of inference but of various people who are informed and rational to different degrees. Inference about inference, together with jumping out of the system [9, 9], characterises systems consisting of human beings. Although our mathematical analysis and simulations are far from perfect, we hope that our analysis shed a new light on the analysis of Schredelseker's model, which could stand for further analysis and laboratory experiments as a test bed for the theory of inter-inference about inter-inference.

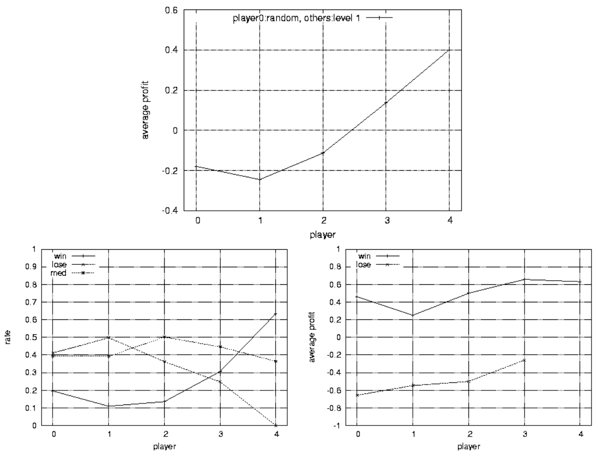


Figure 3. The case where Player Zero chooses R_1 randomly between $Q^{\min} = 0$ and $Q^{\max} = 4$ while the other players follow their level one strategies (the number of players is five).
Upper graph. Although Player Zero's loss is smaller than it is when he follows his level one strategy (compare Figure 1), it is still negative. Conjecture (C) mentioned in Introduction is wrong, but (B) is not realised. The reason Player Zero suffers from losses is twofold: (i) the absolute value of his positive profit is smaller on average than the absolute value of his loss; (ii) he earns positive profit profit more frequently than he suffers from negative profit. In fact Player Zero's loss comes largely from (ii) at least in the present example.
Lower Right Graph. Player Zero's average profit is certainly smaller than his average loss, but the difference is not very large.
Lower Left Graph. Player Zero loses money more frequently than he wins positive profit. It is because his better informed competitors occupy such advantageous positions that leave him a smaller chance to win and a a larger chance to lose whatever R_0 he may choose. It can be checked in the example of section 4.2. Out of the sixteen possible (y_1, y_2, y_3, y_4), eight offer Player Zero a chance to earn positive profit; four fix his profit zero whatever R_0 he may chooses; four deprive him from enjoying positive profit. In addition, as is apparent in (**??**), R_0 must be in a much smaller interval for $0 < \Pi_0$ than it is for $\Pi_0 < 0$ even in the first eight cases.

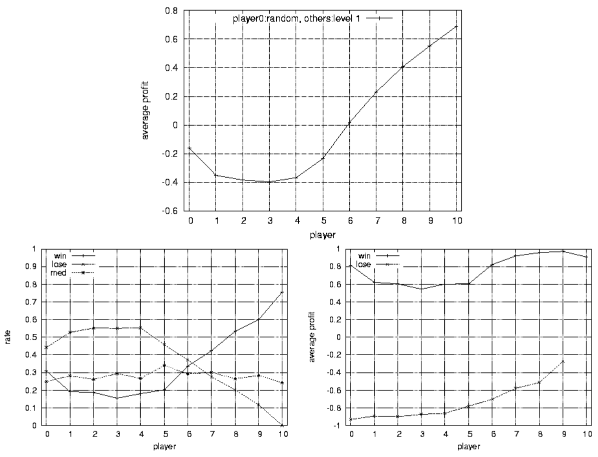


Figure 4. The case where Player Zero chooses R_1 randomly between $Q^{\min} = 0$ and $Q^{\max} = 11$ while the other players follow their level one strategies (the number of players is eleven).

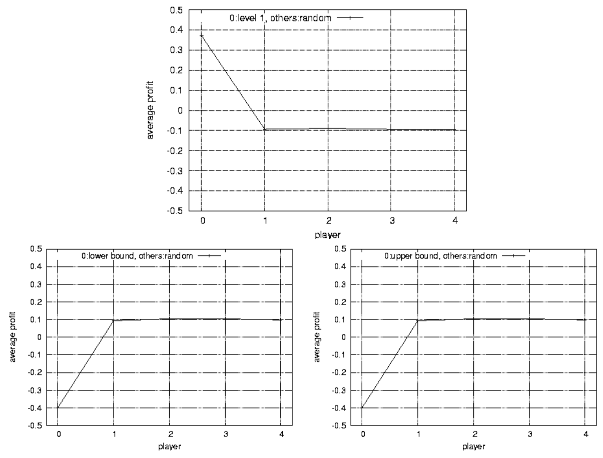


Figure 5. Profit distribution among players (the number of players is five) in the cases where Player 0 chooses $Q_0^{\min} = 0, Q_0^{\star} = 2$ or $Q_0^{\max} = 4$ consistently as R_0 while the other players choose their reservation prices stochastically between 0 and 4 with equal probability. If his competitors follow their level one strategies, both $R_0 = 0$ and $R_0 = 4$ bring about larger profit in the long run than $R_0 = 2$: as (**??**) shows, $R_0^{\star} = -0.125$ for $R_0 = 0$ or $R_0 = 4$ while $R_0^{\star} = -0.25$ for $R_0 = 2$. As is shown above, however, if her rivals determine their reservation prices randomly, $R_0 = 2$ is much more profitable for her than $R_0 = 0$ and $R_0 = 4$ are.
In addition it is observed in our simulations for $1 \leq k$ and $3 \leq N$ that $R_k = Q^{\star}(\boldsymbol{x}_k)$ brings about greater $\Pi_k^{\star}$ than $R_k = Q^{\min}(\boldsymbol{x}_k)$ and $R_k = Q^{\max}(\boldsymbol{x}_k)$ do if other players all choose their reservation prices randomly. It seems to suggest that $\mathbf{L1}_k$ may be the strategy that maximises Π_k if all the other players choose their reservation prices randomly, though rigorous mathematical proof is wanted.

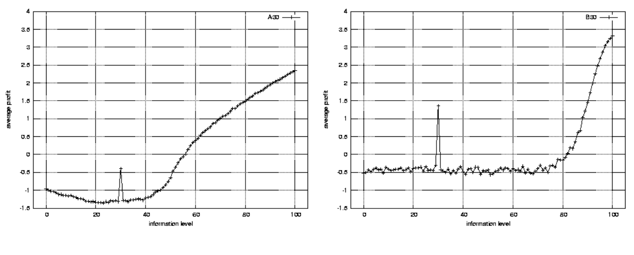


Figure 6. Left Graph. Income distribution among a human subject (Player 30) and middle-value strategy agents. **Right Graph.** Income distribution among a human subject (Player 30) and either-end strategy agents

REFERENCES

[1] Teck-Hua Ho, Colin Camerer and Kerth Weigelt (1998): "Iterated Dominance and Iterated Best Response i Experimental p-Beauty Contest" in *American Economic Review*, vol. 88, pp. 947-969.

[2] Jüergen Huber, Michael Kirchler and Matthias Sutter (2008): "Is More Information Always Better? Experimental Financial Markets With Cumulative Information", in *Journal of Economic Behavior and Organization*, vol. 65, pp. 86-104.

[3] John Maynard Keynes (1973): *The Collected Writings of John Meynard Keynes Volume Seven: The General Theory of Employment, Interest and Money*, Macmillan.

[4] H. Maulin (1986): *Game Theory for Social Sciences*, New York University Press.

[5] R. Nagel (1995): "Unraveling in Guessing Games: An Experimental Study" in *American Economic Review*, vol. 85, pp. 1313-1326.

[6] Sobei H. Oda and Gen Masumoto (2008): "A Note on Snhredlsekr's Maodel: Porivate Innfotmttio and Infernce about Inference", to appear in *Information, Interaction, and (In)Efficiency in Financial Markets*, edited by Huber, J. and Hanke, M., pp. 37-61, Linde Verlag, Vienna, Austria.

[7] Kraus Schredelseker (2007): "Financial Analysis: What is it good for?" *mimeo*.

[8] Mariko Yasugi and Sobei H. Oda (2002): "A Note on the Wise Girls Puzzle" in *Economic Theory*, vol. 19 (no. 1), pp. 145-156.

[9] Mariko Yasugi and Sobei H. Oda (2003): "Notes on Bounded Rationality" in *Scientiae Mathematicae Japonicae*, vol. 57 (no. 1), pp. 83–92.

[10] Hiroyasu Yoneda, Gen Masumoto and Sobei H. Oda (2007): "How to Use Private Information in a Muti-person Zero-sum Game" in *Developments on Experimental Economics* edited by Sobei H. Oda, Springer-Verlag, pp. 239-244.

Addressing NP-Complete Puzzles with Monte-Carlo Methods[1]

Maarten P.D. Schadd and **Mark H.M. Winands**
H. Jaap van den Herik and **Huib Aldewereld**[2]

Abstract.
NP-complete problems are a challenging task for researchers, who investigate tractable versions and attempt to generalise the methods used for solving them. Over the years a large set of successful standard methods have been developed. We mention A* and IDA* which have proven to be reasonably successful in solving a set of NP-complete problems, particularly single-agent games (puzzles). However, sometimes these methods do not work well. The intriguing question then is whether there are new methods that will help us out.

In this paper we investigate whether Monte-Carlo Tree-Search (MCTS) is an interesting alternative. We propose a new MCTS variant, called Single-Player Monte-Carlo Tree-Search (SP-MCTS). Our domain of research is the puzzle SameGame. It turned out that our SP-MCTS program gained the highest scores so far on the standardised test set. So, SP-MCTS can be considered as a new method for addressing NP-complete puzzles successfully.

1 INTRODUCTION

Creating and improving solvers for tractable versions of NP-complete problems is a challenging task in the field of Artificial Intelligence research. As Cook [9] proved: all problems in the class of NP-complete problems are translatable to one another [16]. This implies that a solution procedure for one problem also holds for other problems. Otherwise stated: if an effective method is found to solve a particular instance of a problem, many other problems may be solved as well using the same method.

Games are often NP-complete problems. The rules for games are well-defined and it is easy to compare different methods. For our investigations we have chosen a one-person perfect-information game (a puzzle[3]) called $SameGame$. In Section 2 we will prove that this puzzle is NP-complete.

The traditional methods for solving puzzles, such as the 15×15 puzzle and Sokoban, are A* [15] or IDA* [19]. Other problems, such as the Travelling Salesman Problem (TSP) [3] require different methods (e.g., Simulated Annealing [12] or Neural Networks [23]). These methods have been shown to solve the puzzles mentioned above reasonably well. An example of a practical and successful use of these methods are pathfinders which are, for example, used inside an increasing number of cars. A drawback of the methods is that they need an admissible heuristic evaluation function. The construction of such a function may be difficult.

An alternative to these methods can be found in Monte-Carlo Tree Search (MCTS) [7, 10, 18] because it does not need an admissible heuristic. Especially in the game of Go, which has a large search space [5], MCTS methods have proven to be successful [7, 10]. In this paper we will investigate how MCTS addresses NP-complete puzzles. For this purpose, we introduce a new MCTS variant called SP-MCTS.

The course of the paper is as follows. In Section 2 we present the background and rules of SameGame. Also, we prove that SameGame is NP-complete. In Section 3 we discuss why classical methods are not suitable for SameGame. Then we introduce our SP-MCTS approach in Section 4. Experiments and results are given in Section 5. Section 6 shows our conclusions and indicates future research.

2 SAMEGAME

We start by presenting some background information on SameGame in Subsection 2.1. Subsequently we explain the rules in Subsection 2.2. Finally, we prove that SameGame is NP-complete in Subsection 2.3.

2.1 Background

SameGame is a puzzle invented by Kuniaki Moribe under the name *Chain Shot!* in 1985. It was distributed for Fujitsu FM-8/7 series in a monthly personal computer magazine called *Gekkan ASCII* [20]. The puzzle was afterwards re-created by Eiji Fukumoto under the name of *SameGame* in 1992. So far, the best program for SameGame has been developed by Billings [24].

2.2 Rules

SameGame is played on a rectangular vertically-placed 15×15 board initially filled with blocks of 5 colours at random. A move consists of removing a group of (at least two) orthogonally adjacent blocks of the same colour. The blocks on top of the removed group will fall down. As soon as empty columns occur, the columns to the right are shifted to the left. For each removed group points are rewarded. The amount of points is dependent on the number of blocks removed and can be computed by the formula $(n-2)^2$, where n is the size of the removed group.

We show two example moves in Figure 1. When the 'B' group in the third column of position 1(a) is played, it will be removed from the game and the 'C' block on top will fall down, resulting in position

[1] This contribution is a revised version of an article under submission to CG 2008.
[2] Maastricht University, Maastricht, The Netherlands, email: {maarten.schadd, m.winands, herik, h.aldewereld}@micc.unimaas.nl
[3] Although arbitrary, we will call these one-player games with perfect information for the sake of brevity *puzzles*.

1(b). Because of this move, it is now possible to remove a large group of ‘C’ blocks (n=5). Owing to an empty column the two columns at the right side of the board are shifted to the left, resulting in position 1(c).[4] The first move is worth 0 points; the second move is worth 9 points.

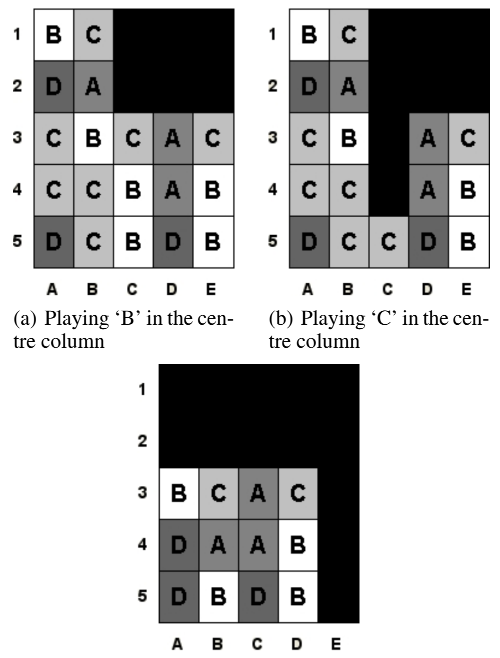

(a) Playing ‘B’ in the centre column
(b) Playing ‘C’ in the centre column
(c) Resulting position

Figure 1. Example SameGame moves.

The game is over if either the player (1) has removed all blocks or (2) is left with a position where no adjacent blocks have the same colour. In the first case, 1,000 bonus points are rewarded. In the second case, points will be deducted. The formula for deduction is similar to the formula for rewarding points but is now iteratively applied for each colour left on the board. During deduction it is assumed that all blocks of the same colour are connected.

There are variations that differ in board size and the number of colours, but the 15×15 variant with 5 colours is the accepted standard. If a variant differs in scoring function, it is named differently (e.g., Jawbreaker, Clickomania) [2, 21].

2.3 Complexity of SameGame

The complexity of a game indicates a measure of hardness of solving the game. Two important measurements for the complexity of a game are the game-tree complexity and the state-space complexity [1]. The game-tree complexity is an estimation of the number of leaf nodes that the complete search tree would contain to solve the initial position. The state-space complexity indicates the total number of possible states.

For SameGame these complexities are as follows. The game-tree complexity can be approximated by simulation. For SameGame, the game-tree complexity for a random initial position is 10^{85} in average. The state-space complexity is computed rather straightforwardly. It is possible to calculate the number of combinations for one column by $C = \sum_{n=0}^{r} c^n$ where r is the height of the column and c is the number of colours. To compute the state-space complexity we take C^k where k is the number of columns. For SameGame we have 10^{159} states. This is not the exact number because a small percentage of the positions are symmetrical.

Furthermore, the hardness of a game can be described by deciding to which complexity class it belongs [16]. The similar game Clickomania was proven to be NP-complete by [2]. However, the complexity of SameGame can be different. The more points are rewarded for removing large groups, the more the characteristics of the game differ from Clickomania. In Clickomania the only goal is to remove as many blocks as possible, whereas in SameGame points are rewarded for removing large groups as well. In the following, we prove that SameGame independently from its evaluation function belongs to the class of NP-complete problems, such as the 3-SAT problem [9].

Theorem 1 *SameGame is NP-complete*

For a proof that it is NP-complete, it is sufficient to reduce SameGame to a simpler problem. We reduce SameGame to Clickomania, which has been proven to be NP-complete with 5 colours and 2 Columns [2]. A SameGame instance with 2 columns is easier to solve than the standard SameGame instance with 15 columns. Instead of proving that finding the optimal path is NP-complete, we prove that checking whether a solution s is optimal is already NP-complete. A solution is a path from the initial position to a terminal position. Either s (1) has removed all blocks from the game or (2) has finished with blocks remaining on the board. Even in the second case a search has to be performed to investigate whether a solution exists that clears the board and improves the score. If we prove that searching all solutions which clear the board is NP-complete, then SameGame is NP-complete as well.

Clickomania is a variant of SameGame where no points are rewarded and the only objective is to clear the board. Finding one solution to this problem is easier than finding every solution. Therefore, it is proven that SameGame is a harder problem than Clickomania; SameGame is NP-complete, too.

3 CLASSICAL METHODS: A* AND IDA*

The classical approach to puzzles involves methods such as A* [15] and IDA* [19]. A* is a best-first search where all nodes have to be stored in a list. The list is sorted by an admissible evaluation function. At each iteration the first element is removed from the list and its children are added to the sorted list. This process is continued until the goal state arrives at the start of the list.

IDA* is an iterative deepening variant of A* search. It uses a depth-first approach in such a way that there is no need to store the complete tree in memory. The search will continue depth-first until the cost of arriving at a leaf node and the value of the evaluation function pass a certain threshold. When the search returns without a result, the threshold is increased.

Both methods are heavily dependent on the quality of the evaluation function. Even if the function is an admissible under-estimator, it still has to give an accurate estimation. Well-known puzzles where this approach works well are the Eight Puzzle with its larger relatives [19, 22] and Sokoban [17]. Here a good under-estimator is the well-known Manhattan Distance. The main task in this field of research is to improve the evaluation function, e.g., with pattern databases [11, 13].

[4] Shifting the columns at the left side to the right would not have made a difference in points. For consistency, we will always shift columns to the left.

These classical methods fail for SameGame because it is not easy to make an admissible under-estimator that still gives an accurate estimation. An attempt to make such an evaluation function is by just rewarding points to the groups on the board without actually playing a move. However, if an optimal solution to a SameGame problem has to be found, we may argue that an "over-estimator" of the position is needed. An admissible over-estimator can be created by assuming that all blocks of the same colour are connected and would be able to be removed at once. This function can be improved by checking whether there is a colour with only one block remaining on the board. If this is the case, the 1,000 bonus points at the end can be deducted. However, such an evaluation function is far from the real score on a position and does not give good results with A* and IDA*. Tests have shown that using A* and IDA* with the proposed "over-estimator" resemble a simple breadth-first search. The problem is that after expanding a node, the heuristic value of a child is significantly lower than the value of its parent, unless a move removes all blocks with one colour from the board.

Since no good evaluation function has been found yet, SameGame presents a new challenge for the puzzle research. In the next section we will discuss our SP-MCTS method.

4 MONTE-CARLO TREE SEARCH

This section first gives a description of SP-MCTS in Subsection 4.1. Thereafter we will explain the Meta-Search extension in Subsection 4.2.

4.1 SP-MCTS

MCTS is a best-first search method, which does not require a positional evaluation function. MCTS builds a search tree employing Monte-Carlo evaluations at the leaf nodes. Each node in the tree represents an actual board position and typically stores the average score found in the corresponding subtree and the number of visits. MCTS constitutes a family of tree-search algorithms applicable to the domain of board games [7, 10, 18].

In general, MCTS consists of four steps, repeated until time has run out [8]. (1) A *selection strategy* is used for traversing the tree from the root to a leaf. (2) A *simulation strategy* is used to finish the game starting from the leaf node of the search tree. (3) The *expansion strategy* is used to determine how many and which children are stored as promising leaf nodes in the tree. (4) Finally, the result of the MC evaluation is propagated backwards to the root using a *back-propagation strategy*.

Based on MCTS, we propose an adapted version for puzzles: Single-Player Monte-Carlo Tree Search (SP-MCTS). Below, we will discuss the four corresponding phases and point out differences between SP-MCTS and MCTS.

Selection Strategy Selection is the strategic task that selects one of the children of a given node. It controls the balance between $exploitation$ and $exploration$. Exploitation is the task to focus on the move that led to the best results so far. Exploration deals with the less promising moves that still may have to be explored, due to the uncertainty of their evaluation so far. In MCTS at each node starting from the root a child has to be selected until a leaf node is reached. Several algorithms have been designed for this setup [7, 10].

Kocsis and Szepesvári [18] proposed the selection strategy UCT (Upper Confidence bounds applied to Trees). For SP-MCTS, we use a modified UCT version. At the selection of node N with children N_i, the strategy chooses the move, which maximises the following formula.

$$\overline{X} \quad + \quad C \cdot \sqrt{\frac{ln\ t\,(N)}{t\,(N_i)}} \quad + \quad \sqrt{\frac{\sum x^2 - t\,(N_i) \cdot \overline{X}^2 + D}{t\,(N_i)}}. \quad (1)$$

The first two terms constitute the original UCT formula. It uses the number of times $t\,(N)$ that node N was visited and the number of times $t\,(N_i)$ that child N_i was visited to give an upper confidence bound for the average game value $\overline{X}$. For puzzles, we added a third term, which represents the deviation [10, 6]. This term makes sure that nodes, which have been rarely explored, are not under-estimated. $\sum x^2$ is the sum of the squared results achieved in this node so far. The third term can be tuned by the constant D. Coulom [10] chooses a move according to the selection strategy only if $t\,(N_i)$ reached a certain threshold (here 10). Before that happens, the simulation strategy is used, which will be explained later. Below we describe two differences between puzzles and two-player games, which may affect the selection strategy.

First, the essential difference between the two is the *range of values*. In two-player games, the results of a game can be summarised by *loss*, *draw*, and *win*. They can be expressed as numbers from the set $\{-1, 0, 1\}$. The average score of a node will always stay in $[-1,1]$. In a puzzle, a certain score can be achieved that is outside this interval. In SameGame there are positions, which can be finished with a value above 4,000 points. If the maximum score for a position would be known, then it is possible to scale this value back into the mentioned interval. However, the maximum score of a position might not be known. Thus, much higher values for the constants C and D have to be chosen than is usual in two-player games.

A second difference for puzzles is that there is *no uncertainty on the opponent's play*. This means that solely the line of play has to be optimised regarding the top score and not the average of a subtree.

Simulation Strategy Starting from a leaf node, random moves are played until the end of the game. In order to improve the quality of the games, the moves are chosen pseudo-randomly based on heuristic knowledge.

In SameGame, we have designed two static simulation strategies. We named these strategies "TabuRandom" and "TabuColourRandom". Both strategies aim at making large groups of one colour. In SameGame, making large groups of blocks is advantageous.

"TabuRandom" chooses a random colour at the start of a simulation. It is not allowed to play this colour during the random simulations unless there are no other moves possible. With this strategy large groups of the chosen colour will be formed automatically.

The new aspect in the "TabuColourRandom" with respect to the previous strategy is that the chosen colour is the colour most frequently occurring at the start of the simulation. This may increase the probability of having large groups during the random simulation.

Expansion Strategy The expansion strategy decides which nodes are added to the tree. Coulom [10] proposed to expand one child per simulation. With his strategy, the expanded node corresponds to the first encountered position that was not present in the tree. This is also the strategy we used for SameGame.

Back-Propagation Strategy During the back-propagation phase, the result of the simulation at the leaf node is propagated backwards to the root. Several back-propagation strategies have been proposed in the literature [7, 10]. The best results that we have obtained was by using the plain average of the simulations. Therefore, we update (1) the average score of a node. Additional to this, we also update (2) the sum of the squared results because of the third term in the selection strategy (see Formula 1), and (3) the best score achieved so far for computational reasons.

The four phases are iterated until the time runs out.[5] When this happens, a final move selection is used to determine, which move should be played. In two-player games (with an analogous run-out-of-time procedure) the best move according to this strategy will be played by the player to move and the opponent then has time to calculate his response. But in puzzles this can be done differently. In puzzles it is not needed to wait for an unknown reply of an opponent. Because of this, it is possible to perform one large search from the initial position and then play all moves at once. With this approach all moves at the start are under consideration until the time for SP-MCTS runs out.

4.2 Meta-Search

A Meta-Search is a search method that does not perform a search on its own but uses other search processes to arrive at an answer. For instance, Gomes *et al.* [14] proposed a form of iterative deepening to handle heavy-tailed scheduling tasks. The problem was that the search was lost in a large subtree, which would take a large amount of time to perform, while there are shallow answers in other parts of the tree. The possibility exists that by restarting the search a different part of the tree was searched with an easy answer.

We discovered that it is important to generate deep trees in SameGame (see Section 5.2). However, by exploiting the most-promising lines of play, the SP-MCTS can be caught in local maxima. So, we extended SP-MCTS with a straightforward form of Meta-Search to overcome this problem. After a certain amount of time, SP-MCTS just restarts the search with a different random seed. The best path returned at the end of the Meta-Search is the path with the highest score found in the searches. Section 5.3 shows that this form of Meta-Search is able to increase the average score significantly.

5 EXPERIMENTS AND RESULTS

Subsection 5.1 shows tests of the quality of the two simulation strategies TabuRandom and TabuColourRandom. Thereafter, the results of the parameter tuning are presented in Subsection 5.2. Next, in Subsection 5.3 the performance of the Meta-Search on a set of 250 positions is shown. Finally, Subsection 5.4 compares SP-MCTS to IDA* and Depth-Budgeted Search (used in the program by Billings [4]).

5.1 Simulation Strategy

In order to test the effectiveness of the two simulation strategies we used a test set of 250 randomly generated positions.[6] We applied SP-MCTS without the Meta-Search extension for each position until 10 million nodes were reached in memory. These runs typically take 5 to 6 minutes per position. The best score found during the search is the final score for the position. The constants C and D were set to 0.5 and 10,000, respectively. The results are shown in Table 1.

Table 1 shows that the TabuRandom strategy has a significant better average score (i.e., 700 points) than plain random. Using the TabuColourRandom strategy the average score is increased by another 300 points. We observe that a low standard deviation is achieved for the random strategy. In this case, it implies that all positions score almost equally low.

Table 1. Effectiveness of the simulation strategies

	Average Score	StDev
Random	2,069	322
TabuRandom	2,737	445
TabuColourRandom	3,038	479

5.2 SP-MCTS Parameter Tuning

This subsection presents the parameter tuning in SP-MCTS. Three different settings were used for the pair of constants $(C; D)$ of Formula 1, in order to investigate which balance between exploitation and exploration gives the best results. These constants were tested with three different time controls on the test set of 250 positions, expressed by a maximum number of nodes. The three numbers are 10^5, 10^6 and 5×10^6. The short time control refers to a run with a maximum of 10^5 nodes in memory. In the medium time control, 10^6 nodes are allowed in memory, and in long time control 5×10^6 nodes are allowed. We have chosen to use nodes in memory as measurement to keep the results hardware-independent. The parameter pair (0.1; 32) represents $exploitation$, (1; 20,000) performs $exploration$, and (0.5; 10,000) is a balance between the other two.

Table 2 shows the performance of the SP-MCTS approach for the three time controls. The short time control corresponds to approximately 20 seconds per position. The best results are achieved by exploitation. The score is 2,552. With this setting the search is able to build trees that have on average the deepest leaf node at ply 63, implying that a substantial part of the chosen line of play is inside the SP-MCTS tree. Also, we see that the other two settings are not generating a deep tree.

In the medium time control, the best results were achieved by using the balanced setting. It scores 2,858 points. Moreover, Table 2 showed that the average score of the balanced setting increased most compared to the short time control, viz. 470. The balanced setting is now able to build substantially deeper trees than in the short time control (37 vs. 19). An interesting observation can be made by comparing the score of the exploration setting in the medium time control to the exploitation score in the short time control. Even with 10 times the amount of time, exploring is not able to achieve a significantly higher score than exploiting.

The results for the long experiment are that the balanced setting again achieves the highest score with 3,008 points. Now its deepest node on average is at ply 59. However, the exploitation setting only scores 200 points fewer than the balanced setting and 100 fewer than exploration.

From the results presented we may draw two conclusions. First we may conclude that it is important to have a deep search tree. Second, exploiting local maxima can be more advantageous than searching for the global maxima when the search only has a small amount of time.

[5] In general, there is no time limitation for puzzles. However, a time limit is necessary to make testing possible.

[6] The test set can be found online at http://www.cs.unimaas.nl/maarten.schadd/SameGame/TestSet.txt

Table 2. Results of SP-MCTS for different settings

10^5 nodes	Exploitation (0.1; 32)	Balanced (0.5; 10,000)	Exploration (1; 20,000)
Average Score	2,552	2,388	2,197
Standard Deviation	572	501	450
Average Depth	25	7	3
Average Deepest Node	63	19	8
10^6 nodes	(0.1; 32)	(0.5; 10,000)	(1; 20,000)
Average Score	2,674	2,858	2,579
Standard Deviation	607	560	492
Average Depth	36	14	6
Average Deepest Node	71	37	15
5×10^6 nodes	(0.1; 32)	(0.5; 10,000)	(1; 20,000)
Average Score	2,806	3,008	2,901
Standard Deviation	576	524	518
Average Depth	40	18	9
Average Deepest Node	69	59	20

5.3 Meta-Search

This section presents the performance tests of the Meta-Search extension of SP-MCTS on the set of 250 positions. We remark that the experiments are time constrained. Each experiment could only use 5×10^5 nodes in total and the Meta-Search distributed these nodes fairly among the number of runs. It means that a single run can take all 5×10^5 nodes, but that two runs would only use 250,000 nodes each. We used the exploitation setting (0.1; 32) for this experiment. The results are depicted in Figure 2.

Figure 2 indicates that already with two runs instead of one, a significant performance increase of 140 points is achieved. Furthermore, the maximum average score of the Meta-Search is at ten runs, which uses 50,000 nodes for each run. Here, the average score is 2,970 points. This result is almost as good as the best score found in Table 2, but with the difference that the Meta-Search used one tenth of the number of nodes. After ten runs the performance decreases because the generated trees are not deep enough.

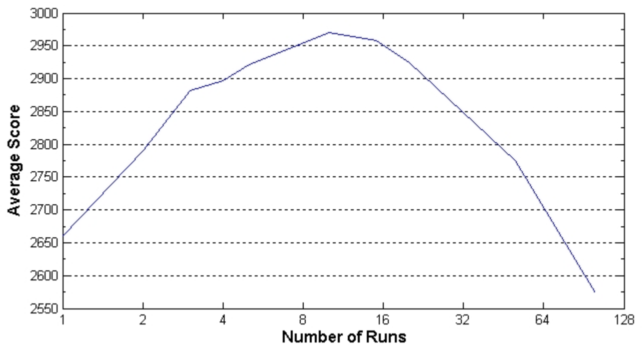


Figure 2. The average score for different settings of the Meta-Search

5.4 Comparison

The best SameGame program so far has been written by Billings [4]. This program performs a non-documented method called Depth-Budgeted Search (DBS). When the search reaches a depth where its budget has been spent, a greedy simulation is performed. On a standardised test set of 20 positions[7] his program achieved a total score of 72,816 points with 2 to 3 hours computing time per position. Using the same time control, we tested SP-MCTS on this set. We used again the exploitation setting (0.1; 32) and the Meta-Search extension, which applied 1,000 runs using 100,000 nodes for each search process. For assessment, we tested IDA* using the evaluation function described in Section 3. Table 3 compares IDA*, DBS, and SP-MCTS with each other.

Table 3. Comparing the scores on the standardised test set

Position nr.	IDA*	DBS	SP-MCTS
1	548	2,061	2,557
2	1,042	3,513	3,749
3	841	3,151	3,085
4	1,355	3,653	3,641
5	1,012	3,093	3,653
6	843	4,101	3,971
7	1,250	2,507	2,797
8	1,246	3,819	3,715
9	1,887	4,649	4,603
10	668	3,199	3,213
11	1,073	2,911	3,047
12	602	2,979	3,131
13	667	3,209	3,097
14	749	2,685	2,859
15	745	3,259	3,183
16	1,647	4,765	4,879
17	1,284	4,447	4,609
18	2,586	5,099	4,853
19	1,437	4,865	4,503
20	872	4,851	4,853
Total:	22,354	72,816	73,998

[7] The positions can be found at the following address: http://www.js-games.de/eng/games/samegame.

SP-MCTS outperformed DBS on 11 of the 20 positions and was able to achieve a total score of 73,998. Furthermore, Table 3 shows that IDA* does not perform well for this puzzle. It plays at the human beginner level. The best variants discovered by SP-MCTS can be found on our website.[8] There we see that SP-MCTS is able to clear the board for all of the 20 positions. This confirms that a deep search tree is important for SameGame as was seen in Subsection 5.2.

By combining the scores of DBS and SP-MCTS we computed that at least 75,152 points can be achieved for this set.

6 CONCLUSIONS AND FUTURE RESEARCH

In this paper we have shown how MCTS addresses NP-complete puzzles. As a representative puzzle, we have chosen the game SameGame and have proven that it is NP-complete. We proposed a new MCTS variant called Single-Player Monte-Carlo Tree Search (SP-MCTS) as an alternative to more classical approaches that solve (NP-complete) puzzles, such as A* and IDA*. We adapted MCTS by two modifications resulting in SP-MCTS. The modifications are (1) the selection strategy and (2) the back-propagation strategy. Below we provide three observations and subsequently two conclusions.

6.1 Conclusions

First, we observed that our TabuColourRandom strategy (i.e., reserving the most frequent occurring colour to be played last) significantly increased the score of the random simulations in SameGame. Compared to the pure random simulations, an increase of 50% in the average score is achieved.

Next, we observed that it is important to build deep SP-MCTS trees. Exploiting works better than exploring at short time controls. At longer time controls the balanced setting achieves the highest score, and the exploration setting works better than the exploitation setting. However, exploiting the local maxima still leads to comparable high scores.

Third, with respect to the extended SP-MCTS endowed with a straightforward Meta-Search, we observed that for SameGame combining a large number of small searches can be more beneficial than doing one large search.

From the results of SP-MCTS with parameters (0.1; 32) and with Meta-Search set on a time control of around 2 hours we may conclude that SP-MCTS produced the highest score found so far for the standardised test set. It was able to achieve 73,998 points, breaking Billings' record by 1,182 points. So, our program with SP-MCTS may be considered at this moment the world's best SameGame program.

A second conclusion is that we have shown that SP-MCTS is applicable to a one-person perfect-information game. SP-MCTS is able to achieve good results on the NP-complete game of SameGame. This means that SP-MCTS is a worthy alternative for puzzles where a good admissible estimator cannot be found. Even more, SP-MCTS proves to be an interesting solution to solving similar tractable instances of NP-complete problems.

6.2 Future Research

In the future, more enhanced methods will be tested on SameGame. We mention three of them. First, knowledge can be included in the selection mechanism. A method to achieve this is called *Progressive Unpruning* [8]. Second, this paper demonstrated that combining small searches can achieve better scores than one large search. However, there is no information shared between the searches. This can be achieved by using a transposition table, which is not cleared at the end of a small search. Third, the Meta-Search can be parallelised asynchronously to take advantage of multi-processor architectures.

Furthermore, to test our theories about the successfulness of SP-MCTS in solving other NP-Complete problems, we would like to investigate how well this method performs on, for instance, (3-) SAT problems.

[8] The best variations can be found at the following address: http://www.cs.unimaas.nl/maarten.schadd/SameGame/Solutions.html

ACKNOWLEDGEMENTS

This work is funded by the Dutch Organisation for Scientific Research (NWO) in the framework of the project TACTICS, grant number 612.000.525.

REFERENCES

[1] L. V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Rijksuniversiteit Limburg, Maastricht, The Netherlands, 1994.

[2] T. C. Biedl, E. D. Demaine, M. L. Demaine, R. Fleischer, L. Jacobsen, and I. Munro. The Complexity of Clickomania. In R. J. Nowakowski, editor, *More Games of No Chance, Proc. MSRI Workshop on Combinatorial Games*, pages 389–404, MSRI Publ., Berkeley, CA, Cambridge University Press, Cambridge, 2002.

[3] N. L. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory 1736-1936*. Clarendon Press, Oxford, UK, 1976.

[4] D. Billings. Personal Communication, University of Alberta, Canada, 2007.

[5] B. Bouzy and T. Cazanave. Computer Go: An AI-Oriented Survey. *Artificial Intelligence*, 132(1):39–103, 2001.

[6] G. M. J. B. Chaslot, S. De Jong, J-T. Saito, and J. W. H. M. Uiterwijk. Monte-Carlo Tree Search in Production Management Problems. In P. Y. Schobbens, W. Vanhoof, and G. Schwanen, editors, *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence*, pages 91–98, Namur, Belgium, 2006.

[7] G. M. J. B. Chaslot, J-T. Saito, B. Bouzy, J. W. H. M. Uiterwijk, and H. J. van den Herik. Monte-Carlo Strategies for Computer Go. In P. Y. Schobbens, W. Vanhoof, and G. Schwanen, editors, *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence*, pages 83–91, Namur, Belgium, 2006.

[8] G. M. J. B. Chaslot, M. H. M. Winands, J. W. H. M. Uiterwijk, H. J. van den Herik, and B. Bouzy. Progressive strategies for Monte-Carlo Tree Search. In P. Wang et al., editors, *Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007)*, pages 655–661. World Scientific Publishing Co. Pte. Ltd., 2007.

[9] S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[10] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, editors, *Proceedings of the 5th International Conference on Computer and Games*, volume 4630 of *Lecture Notes in Computer Science (LNCS)*, pages 72–83. Springer-Verlag, Heidelberg, Germany, 2007.

[11] J. C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[12] R. W. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46(3):271–281, 1990.

[13] A. Felner, U. Zahavi, Jonathan Schaeffer, and R. C. Holte. Dual Lookups in Pattern Databases. In *IJCAI*, pages 103–108, Edinburgh, Scotland, UK, 2005.

[14] C. P. Gomes, B. Selman, K. McAloon, and C. Tretkoff. Randomization in Backtrack Search: Exploiting Heavy-Tailed Profiles for Solving Hard Scheduling Problems. In *AIPS*, pages 208–213, Pittsburg, PA, 1998.

[15] P. E. Hart, N. J. Nielson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernatics*, SSC-4(2):100–107, 1968.

[16] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 67–161. 1990.

[17] A. Junghanns. *Pushing the Limits: New Developments in Single Agent Search*. PhD thesis, University of Alberta, Alberta, Canada, 1999.

[18] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the EMCL 2006*, volume 4212 of *Lecture Notes in Computer Science (LNCS)*, pages 282–293, Berlin, 2006. Springer-Verlag, Heidelberg, Germany.

[19] R. E. Korf. Depth-first iterative deepening: An optimal admissable tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[20] K. Moribe. Chain shot! *Gekkan ASCII*, (November issue), 1985. (In Japanese).

[21] PDA Game Guide.com. Pocket PC Jawbreaker Game. *The Ultimate Guide to PDA Games*, Retrieved 7.1.2008. http://www.pdagameguide.com/jawbreaker-game.html.

[22] A. Sadikov and I. Bratko. Solving 20 $\times$ 20 Puzzles. In H. J. van den Herik, J. W. H. M. Uiterwijk, M. H. M. Winands, and M. P. D. Schadd, editors, *Proceedings of the Computer Games Workshop 2007 (CGW 2007)*, pages 157–164, Universiteit Maastricht, Maastricht, The Netherlands, 2007.

[23] J. J. Schneider and S. Kirkpatrick. *Stochastic Optimization*, Chapter Application of Neural Networks to TSP, pages 405–413. Springer-Verlag, Berlin Heidelberg, Germany, 2006.

[24] University of Alberta GAMES Group. GAMES Group News (Archives), 2002. http://www.cs.ualberta.ca/ games/archives.html.

Experimental Computational Philosophy: shedding new lights on (old) philosophical debates

Vincent Wiegel and **Jan van den Berg**[1]

Abstract. Philosophy can benefit from experiments performed in a laboratory for philosophical experimentation (SophoLab). To illustrate the power of Experimental Computational Philosophy, we set up and ran several experiments on a part of Harsanyi's theory on utilitarianism. During decomposition and translation of this theory in the experimental setting of Sopholab, we discovered that it is underspecified. We filled in some blank spots and found out that information and its costs are key in the effectiveness of act and rule utilitarianism. We also identified three further elements that have particular influence on the effectiveness of both strands of utilitarianism: group size of agents, decision-making around uncertainty, and social culture towards particular types of actions. We conclude having shown that setting up computational philosophical experiments is a useful way to gain new and deeper insights in existing argumentations used in old (and new) philosophical debates.

1 Introduction

In philosophy it can be hard to test a theory in practice, i.e., on humans because it would be unethical to expose them to harm or impossible because the number of different settings is simply too large to cover all of them in test situations. In addition, it is often required to analyze a philosophical theory with respect to aspects like consistency, completeness, and soundness of reasoning. These observations invite for the creation of a laboratory where experiments can be set up to test philosophical theories.

As with all experiments, philosophical experiments (should) make use of an environment in which situations of study are reconstructed in a way that abstracts from non-relevant factors. This can be achieved by transforming the theory under examination into a different conceptual structure that exhibits the necessary features of abstraction and control. The new conceptual structure may be constructed by making use of a conceptual framework together with a set of techniques. The role of the conceptual framework is to provide a consistent set of concepts to rephrase the theory. Game theory [5] may be considered as such a conceptual framework, another one is the belief-desire-intention model [1]. Key characteristic of such a framework is that theories that have been formulated in terms of its concepts can easily be prepared for experimental testing by making use of the corresponding techniques. A computer with some software can offer such techniques creating an experimental environment for Computational Philosophy: SophoLab [6]. SophoLab is the name by which the whole of methods, techniques and systems as mentioned above, is designated. The experiments detailed below have been executed using SophoLab.

Several people have worked as a Computational Philosopher. Danielson [2, 3], for example, has constructed computer programs that represent players at a game. He uses the game and the strategies of the players that represent particular moral philosophical stances, to test these positions. In similar ways, computers are increasingly used by social scientists and philosophers to support their research activities. The study of utilitarianism can also benefit from these new means of experimental research because many assumptions and axioms of the theory can be cast in logical and mathematical forms. This motivated us to use the theory of utilitarianism provided by Harsanyi [4] as a test case for experimental computational philosophy. Therefore, the goal of this paper is to introduce the basic ideas underlying experimental computational philosophy and to illustrate the approach by analyzing experimentally Harsanyi's theory of utilitarianism.

The rest of this paper is structured as follows. In section 2 we provide a quick overview of the methodology of experimental computational philosophy. Section 3 prepares the experiments by analyzing Harsanyi's theory of utilitarianism showing some white spots. Section 4 describes the setup and running of the experiments according the methodology described in section 2. This description includes the report of the results found. In the final section 5 we present some conclusions.

2 Experimenting in philosophy

What does it mean to run philosophical experiments? The answer to this question can not be written down in a few statements. Elsewhere, the methodology and the translation are described in more detail [6]. For the current purpose we give a short description of the steps taken in the setting up of experiments, running them, and translating back the results of the experiments. These steps are

1. Decomposing the selected philosophical theory into assumptions, premises, predictions, etc.
2. that can be translated into a concrete experimental setting,
3. and translated into the elements of the intermediate conceptual framework.
4. The concrete experimental setting must be reflected in the intermediate conceptual framework.
5. The theory is implemented in the laboratory based on the requirements of the conceptual framework,
6. reflecting the concrete experimental setting.
7. Experiments are conducted
8. and the results are translated back into the (restated) terms of the theory
9. that can be used to confirm, refine, reject, etc. the theory.

[1] Faculty of Technology, Policy, and Management, Delft University of Technology, The Netherlands, email: {v.wiegel,j.vandenberg}@tudelft.nl

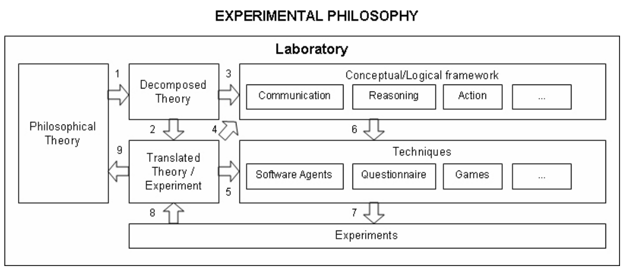


Figure 1. Steps in experimental computational philosophy [6].

As has been announced above, we illustrate the above-given general outline of experimental computational philosophy by analyzing and implementing Harsanyi's theory of utilitarianism. We first start by explaining the relevant basic issues of this theory.

3 Harsanyi's theory of utilitarianism

Harsanyi's theory of utilitarianism [4] has its roots in the work of Adam Smith, Immanuel Kant and the utilitarian tradition of Jeremy Bentham and John Stuart Mill. From Bentham and Mill he takes the concept of maximization of social utility (social welfare function) as the criterion of the morally good. He prefers rule utilitarianism over act utilitarianism and formulates his moral criterion as follows ([4], page 41):

> "...a correct moral rule is that particular behavioral rule that would maximize social utility if it were followed by everybody in all situations of this particular type."

In addition, rationality plays an important role in his theory. In his view ethics is a part of a general theory of rational behavior on par with decision theory and game theory. The rationality criterion is expressed as Pareto optimality and the Bayesian rationality postulates.

Using utilitarianism as our starting point including its definition of what is morally good (i.e., maximum social utility), we focus on the claims that Harsanyi makes as to how best achieve the moral good. In order to be able to test his claims in an experimental setting, we interpret his recommended behavioral rules as strategies as used by (individual) agents. Then, Harsanyi's arguments can be formalized as follows. Let $S_1, S_2, \ldots, S_z$ be the strategies S_i of agents $1, 2, \ldots, z$, where each strategy S_i is an element of the set of all possible strategies. The social welfare function $W(S_1, \ldots, S_z)$ is the sum of all individual utilities:

$$W(S_1, \ldots, S_z) = \sum_{i=1}^{z} U_i(), \tag{1}$$

where $U_i()$ is the utility of agent i. The welfare function $W(S_1, \ldots, S_z)$ is maximized over the strategies S_1 to S_z of all agents:

$$W\text{max} = \max_{S_1 \ldots S_z} W(S_1, \ldots, S_z). \tag{2}$$

The utility function must adhere to the rationality requirements such as a complete pre-ordering and continuity.

In our discussion and experimentation we will focus on Harsanyi's preference of rule over act utilitarianism. We will not be concerned with Harsanyi's assumptions on interpersonal utility comparison, his axiomatic justification of utilitarianism nor with the social welfare function. Hence, we will take many assumptions of his theory for granted whether we agree with them or not. It is our aim to investigate his theory and particular aspects of it, and not argue against it. It is our job as laboratory technician in the SophoLab to set up the experiment with the theory as starting point. The question of rule versus act utilitarianism is the focus point or, more precisely, which version of utilitarianism is preferable. Harsanyi is quite clear on this ([4], page 56):

> "...the basic question we have to ask is this: Which version of utilitarianism will maximize social utility? Will society be better of under one or the other? This test very clearly gives the advantage to rule utilitarianism."

In Harsanyi's view the question of morality is that of maximizing social utility. This is the same for both act and rule utilitarianism. Their decision rule, however, differs. For the rule utilitarian agent the decision of other fellow rule utilitarian agents is an endogenous variable, whereas for the act utilitarian agent the decision of all others, be they utilitarian or otherwise motivated, are exogenous ([4], page 57):

> "An act utilitarian moral agent assumes that the strategies of all other moral agents (including those of all other act utilitarian agents) are given and that his task is merely to choose his own strategy so as to maximize social utility when all other strategies are kept constant. In contrast, a rule utilitarian moral agent will regard not only his own strategy but also the strategies of all other rule utilitarian agents as variables to be determined during the maximization process so as to maximize social utility."

Like has been mentioned above, Harsanyi prefers rule utilitarianism over act utilitarianism. To strengthen this position, Harsanyi gives an elaborate example ([4], pages 57, 58):

> "For example, consider the problem of voting when there is an important measure in the ballot but when voting involves some minor inconvenience. Suppose, there are 1,000 voters strongly favoring the measure, but it can be predicted with reasonable certainty that there will also be 800 negative votes. The measure will pass if it obtains a simple majority of all votes cast. How will the utilitarian theories handle this problem?
> First, suppose that all 1,000 voters favoring the measure are act utilitarian agents. Then each of them will take the trouble to vote only if he thinks that his own vote will be decisive in securing passage of the measure, that is, if he expects exactly 800 other people favoring the meaner to vote (since in this case his own vote will be needed to provide the 801 votes required for majority). But of course, each voter will know that it is extremely unlikely that his own vote will be decisive in this sense. Therefore, most act utilitarian voters will not bother to vote, and the motion will fail (...).
> In contrast, if the 1,000 voters favoring the measure are rule utilitarian agents, then all of them will vote (if mixed strategies are not allowed). This is so because the rule utilitarian decision rule will allow them a choice only between two admissible strategies: one requiring everybody to vote and the other requiring nobody to vote. As this example shows, by following the rule utilitarian decision rule people can achieve successful spontaneous co-operation in situations where this could not be done by adherence to the act utilitarian decision rule (or at least where this could not be done without explicit agreement

on coordinated action, and perhaps without an expensive organization effort)."

However, these statements immediately raise some questions. E.g., based on the assumptions made, rule utilitarian agents seem to be in a better position to organize cooperation and coordinate their strategy. However, we may wonder whether act utilitarian agents are also allowed to cooperate in one way or another. And if so, how does the coordination take place? Is there someone taking the lead and calling on others? What information flow is required for coordination to be successful? What effect would that have on their behavior? Another key issue is that of the decision rules used. Both seek to maximize utility. The main difference is that in the rule utilitarian decision rule the fellow rule utilitarian agents are endogenous, whereas in the act utilitarian decision rule all other agents' strategies are exogenous. Regarding both decision rules the question is for both endogenous and exogenous variables: what assumptions about the values are made? And how? Do the agents form hypotheses about the behavior of others? If so, how do they reason about them? Do they apply some maximum likelihood reasoning?

We hope to have made clear that there are, at first glance, some questions (related to some 'white spots' in the theory) that need answering before we can say that a satisfactory explanation has been given for the claims made. We hypothesize that through the setting up of generic mechanisms to represent the problems, various options can be investigated.

4 The experiments

In this section, we prepare, model and run various experiments related to Harsanyi's theory of utilitarianism. To do so, we apply the steps of the methodology as explained in section 2.

4.1 Step 1: Decomposition

The main components of Harsanyi's rule and act utilitarianism as discussed in the preceding section are

- The utility functions chosen: they will be specified below in Step 2.
- The decision rule for act utilitarian agents favoring more utility over less. This concerns a decision made by each individual act utilitarian agent i yielding a strategy S_i where the agent takes into account the set of (assumed) strategies of all other agents, i.e., $S_{other} = S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_z$.
- The decision rule for the rule utilitarian agents which differs in the definition of the endogenous and exogenous variables. Each rule utilitarian agent uses the same reasoning while taking into the set of (assumed) strategies of all act utilitarian agents: as a consequence of this, all rule utilitarian agents will, if subject to equal circumstances, end up with the same conclusion.
- The social welfare function $W()$ chosen, here defined as the sum of the utilities of the individual agents according to equation (1).

Harsanyi's theory concludes with a preference of rule over act utilitarianism. Therefore, the hypothesis to be tested is the following one: rule utilitarian agents will outperform act utilitarian agents, i.e., $W_{rule} > W_{act}$ where W_{rule}, W_{act} is the social welfare function in case the agents behave like rule utilitarian agents and act utilitarian agents respectively.

4.2 Step 2: Experimental Setting

Step 2 concerns the translation of the problem into a framework and experimental setting. To do so, we take Harsanyi's example: There are two parties, one act and one rule utilitarian party. They are engaged in a voting that each hopes to win. According to the prediction by Harsanyi the rule utilitarian agents will win the vote whenever they have the majority. And more to the point, the hypothesis is that *the act utilitarian agents will not be able to win, even if they have the majority*. Key in Harsanyi's argument is the fact that each (act) utilitarian has two options: to go voting or do something else (that yields a positive utility). As each of act utilitarian agents has to decide for himself, the question he has to answer is: will my voting make any difference? If not, then he will do something else. As each of the act utilitarian agents will think in a similar way none will vote, and the vote is subsequently lost. The act utilitarian agent faces a situation in which there are, logically speaking, four possible outcomes. One, he will not go voting while enough of his fellows will, in which case he derives his share of the benefits from the won vote and the utility of doing X. Two, if the vote is lost he will at least have the utility from action X. Three, if he votes and the vote is lost he will derive utility from neither. Four, if the vote is won, the winning will provide utility but he has forgone the utility associated with X. To set up an executable experiment the utilities and preference functions have to be exact and quantified. We will start using the following pay-off structure defining the individual utility for the act utilitarian agents:

- do something else while enough others votes: 50
- vote while enough others votes: 40
- do something else while not enough others votes: 10
- vote while not enough others votes: 0

The rule utilitarian agents on the other hand will all follow one rule, and if that rule is to vote then they will not be in danger of unintentionally losing the voting game. So, all rule utilitarian agents will always go voting. Therefore, the pay-off structure of the rule utilitarian agents is a simple one: each rule utilitarian agent yields an individual utility of 40 in case the rule utilitarian agents win the voting, and of 0 in case they loose the voting. This discussion points to an open spot in Harsanyi's presentation. Rule utilitarian agents are not able to exploit the majority they have. The surplus of voters cannot be used to gain additional utility doing something else, whereas the act utilitarian agents are able to do so. We finally observe here that the pay-off structures of both types of agents are parameterized and can be changed in the course of the experimentation.

4.3 Steps 3: Translation

The framework is based on the belief desire intention (BDI) model [1], implemented using the technique of computers and Java programming. It consists of agents that represent the moral agents from the Harsanyi example. As in the example they are involved in a voting. They will have the desire to maximize their utility, they holds beliefs about what other agents will do, and they will form intentions to go voting or do something else. They can reason about their beliefs and intentions in a logical way. Trying to translate this in the elements of the intermediate framework, we encounter several problems:

- In Harsanyi's example, the agents know there are 1800 agents involved, of which 800 belong to one party and 1000 to the other party. They know that a simple majority suffices to win the vote.

As how they come to know this and other things, Harsanyi provides no explanation. Actually, Harsanyi assumes that both types of agents have certain information. E.g. with respect to the rule utilitarian agents we observe that (1) all rule utilitarian agents are supposed to be equal (that is adhere to the same decision rule), (2) they know they are the same, (3) they also know about the number of agents involved in the voting, and (4) they know what it takes to win a vote, the qualified majority.

- Perhaps this is fine for the rule utilitarian agents (since, based on these assumptions, they are able to win the voting game in many cases), but for the act utilitarian agents not so assumptions are being made. So several questions may be posed like (1) what information do act utilitarian agents know, (2) how do they get their information from? (3) how do they form their beliefs?, and (4) how do they make decisions given the choice of the other agents. To do their job, they must have some notion about their fellow agents intended behavior. Harsanyi actually does not exclude some kind of agreement or organization to arrive at some form of co-operation. Then the question is: (5) what kind of co-operation would that be? and (6) what will it cost?

From this discussion it should be clear that the argumentation as presented by Harsanyi and the example are not implementable as they are. Most crucial is the absence of an explanation how the utilitarian agents come by some knowledge about the world they live in, who their fellows are and what their intentions are. This forces to a revisiting of the steps 2 and 3.

4.4 Steps 2 till 4 (repeated): Extending and Adjusting

Let us first consider the question where the utilitarian agents get the information about the other agents from. One of the options that poses least demands on institutional organization is having the voters meet in chance groups before the vote, where they can question each other about both their type (rule or act utilitarian) and their intention (voting or not). This requires no pre-organized meetings, no knowledge of the whereabouts of fellows, no deliberate grouping (though that would certainly make sense). Moreover, it seems natural, members of a congress, members of parliament meeting each other in the corridors, the lunch room, etc. discussing the upcoming vote. In addition we observe that information collection might or might not be free. To add the possibility that information is not freely available we introduce a negative utility on enquiry. Agents are further supposed to be free to decide whether or not to acquire information. The utility (cost of information) will be first set at -2 and will be changed during the experiments.

At the start, we assume that the act utilitarian agents will hold no beliefs about other agent's intentions. They all have an innate desire to maximize utility. They go about gathering information in a group with a certain group size. Typically, the groups for the information exchange are much smaller than the voting group (which consists of all potential voters). The information exchange groups are assembled at random. Agents can be added to groups as a member. Agents decide whether they want to participate in the information exchange and voting and sign up for a group. In the experiment, all agents will participate in the information exchange where they ask their fellows about their intentions. Now the question is where does this first intention come from. If each agent asks the other agents about their intention, needing this information to form his own intention, we are stuck. We therefore assume some propensity to go voting. This propensity will be modeled as the chance to go voting (i.e., the complementary chance will express the probability to to do something else). Given the propensity probability, the initial intention to go voting (or not) is assigned randomly to the (act) utilitarian agents. Next, each agent i contacts the members of the information exchange group they belong to, in order to form his (her) personal belief about the expected number $E_i(V)$ of voters of each party (excluding agent's own vote). In its most basic form this is done by extrapolating the findings in the small groups to the whole population.

Given their beliefs, the next question is how the act utilitarian agents will form their final intention (that is transformed into action). To start with, there are three different decision rules that come to mind where α represents the majority vote, i.e., for our voting game with 1800 voters, $\alpha = 901$.

1. Harsanyi's version: if agent i thinks he will make a difference go voting, otherwise do something else or, more formally:

$$\begin{aligned}&\text{Go voting if } \alpha - x < E_i(V) < \alpha + y \text{ where}\\&\quad x = 2 \text{ and } y = 0,\\&\text{otherwise do something else.}\end{aligned} \tag{3}$$

2. Generalized version: if agent i thinks his party will win or lose anyway do not go voting but use the opportunity to gain some additional utility by doing something else or, more formally:

$$\begin{aligned}&\text{Go voting if } \alpha - x < E_i(V) < \alpha + y \text{ where}\\&\quad \alpha - x \in [0, 1800] \text{ and } \alpha + y \in [0, 1800],\\&\text{otherwise do something else.}\end{aligned} \tag{4}$$

3. An even more extended version which is expressed as follows:

$$\begin{aligned}&\text{Stick with intention if } \alpha - x < E_i(V) < \alpha + y,\\&\text{do something else if } E_i(V) \geq \alpha + y,\\&\text{and go voting if } E_i(V) \leq \alpha - x,\\&\quad \text{where } \alpha - x \in [0, 1800] \text{ and } \alpha + y \in [0, 1800].\end{aligned} \tag{5}$$

Rule 1 (defined by (3)) is Harsanyi's version of the act utilitarian decision rule. If, and only if, the agent's vote is decisive, he will go voting. Rule 2 (defined by (4)) is a generalized version of rule 1. The only difference being that the margins within which the agent will conceive his vote as decisive is extended. He does not have to be voter 901 in order to go voting but might be say voter 904. A justification for this extension is uncertainty. Under circumstances it might be hard to get a correct impression of the exact number of voters. One might get the impression wrong by some margin. This margin then has to be taken into account. Rule 3 defined by (5) is different. It is introduced as an alternative in the experiment to see whether other outcomes are possible under different assumptions. It takes the current intention of the agent into account as well as the expected outcome of the voting based on the expected number of voters of each party. If the expectation is that the vote will probably be won, the current intentions by all agents is perfectly suited and should not be altered, including its own one. Again, there are some margins for uncertainty. If the expectation is that the vote will be won by a sufficiently large margin, the agent will decide to do something else. If the expectation is that the vote will be lost this has to be remedied by going to vote. This is probably the hardest element in the decision rule to justify from a rational point of view. The chance that the vote will be decisive is small indeed, the utility of doing something else is guaranteed and should be preferred. Otherwise, if the alternative utility is small, the agent may take the chance and decide to go voting.

4.5 Step 5 and 6: Implementation

Trying to keep the focus on the main arguments has made us decide to skip all kinds of implementation details. Here we only mention that, as basic elements of the SophoLab framework, so-called agents, groups, games, and a referee are used to implement the above-sketched ideas. For more details, we refer to [6], chapter 5.

4.6 Step 7: Running the experiments

Running the experiments consists of executing the computer program with different sets of the parameters. The model embedded in the software has a set of input and output variables. The input variables include the decision rules, the number of runs, the tolerance margins, the number and types of agents, the propensity (to vote or not to vote), the group size (of inquiry groups), the sequence of actions, and the pay-off matrices. The output variables are the average score of all types of agents, as well as their max, min scores.

Table 1. The results of 6 voting runs with 48 act utilitarian and 32 rule utilitarian agents: the first six rows show the input parameter values, the last two rows the resulting output in terms of average score (av sc), maximum score (max sc), and minimum score (min sc).

decision rule	3	number of runs	6
margins	3.5, 3.5	number of agents	80
agent types	act, rule utilitarian	number per type	48, 32
propensity	0.8	inquiry group size	40
sequence	2x inquiry + voting	pay-off inquiry	1
		pay-off voting	50, 40, 10, 0
av sc (act)	243	max, min sc (act)	288, 228
av sc (rule)	0	max, min sc (act)	0, 0

4.6.1 Some first results

Table 1 represents the results of a run in which 80 agents, of which 48 act utilitarian agents and 32 rule utilitarian agents are involved in a voting. The act utilitarian agents follow decision rule 3 (as defined by (5)) with (tolerance) margins of 3.5. This means that if they expect between 37,5 (38) and 44,5 (44) of their fellows to go voting, they will stick to their original intention. The propensity (probability of going to vote, initially) equals 0.8. Gathering information costs 1 utility for act utilitarian agents (not of relevance for rule utilitarian agents) and is done in groups of 40 agents (defined by the inquiry group size), winning the vote brings 40 utilities per agent, etc. The outcomes of this experiment are as follows: on average the act utilitarian have a score of 243 which is slightly more than the rule utilitarian could have achieved. The maximum average utility for the rule utilitarian agents is 240 (6 times 40 for winning the vote when they are the majority party). The act utilitarian that was best had a total utility of 288 while the worst off scored 228.

Following decision rule 2 with wider tolerance margins shows similar results in case the total number of agents is 20, with 12 act utilitarian agents and 8 rule utilitarian agents. The group size in which information is gathered was smaller (namely 5 which equals 25% of the total group size). On average the utility gathered by the act utilitarian agents appeared to be slightly less than what rule utilitarian agents would have achieved. Most importantly, we again observed that the rule utilitarian agents could never win the voting: for further details we refer to [6].

4.6.2 General findings

Many runs were executed in which some variables were kept constant while one or two other variables were varied to investigate its success. By running many such tests a picture arises that we will now describe. The experiments show that, independent of the configuration, decision rule 1 (as described by inequality (3)) is disastrous for the act utilitarian agents. They never win a voting, not even when they form the majority, as predicted by Harsanyi. When the decision rule is relaxed in order to include uncertainty, the act utilitarian agents fare better. In some runs they are able to win the voting. Important seems to be the tolerance in the decision rule, that is the extent of uncertainty allowed for. Decision rule 3 is even more successful. From a fairly small tolerance onwards the act utilitarian agents are able to win the vote when they have the majority. All decision rules allow the act utilitarian agents to exploit the majority surplus. Part of the population does not vote while the vote is still won. In cases where the vote is lost, still some utility is gained by some (rule 2 and 3) or all act utilitarian agents (rule 1).

The tolerance margin can vary from zero to half the size of the population. With a tolerance of zero the decision rule is the one proposed by Harsanyi. With a tolerance of half the population size we have effectively a rule that says ‘vote always’, this is, of course, the rule utilitarian strategy. As Harsanyi predicted with a tolerance of zero, act utilitarian agents are not able to win a vote. What did surprise was that after an increase to about 3.5, act utilitarian agents are almost always winning the vote when they have the majority. Another important element is the cost of information. From the previous aspect of tolerance we learned that some tolerance in the decision making helps. This is, of course, only the case if there is some information about what to expect. Thus information exchange is vital. Information is valuable only if it helps increase the chances of a won vote, which again is in part dependent on the tolerance. As the cost of information increases act utilitarian agents still win their votes, but at an increasing cost. When cost is high rule utilitarian agents do markedly better because they have less need for information. This relationship is directly proportional.

4.7 Steps 8 and 9: Translating back to the theory

The decision rule as described by Harsanyi works out badly for the act utilitarian agents. The generalized version (decision rule 2) works already better while the adapted version (decision rule 3) proves even more beneficial. We argued above that the adaptation of the rule does not violate the act utilitarian character, but does take into account uncertainty (which is left out of Harsanyi’s account). So with a slight relaxation of the decision rule act, utilitarian agents win the vote, contrary to Harsanyi’s prediction. And under certain conditions - larger tolerance margins - we have seen that act utilitarian agents perform better than rule utilitarianism could have done. This follows from their ability to exploit the surplus of votes.

The size of the informal group that exchange information influences the performance significantly. The relationship is not linear. Small (12,5% of total population) and large (50% of total population) groups perform clearly better than medium sized (25% of total population) groups. As the population size grows act utilitarian agents improve their performance. For rule utilitarian agents the minimum and maximum scores are always the same. For the act utilitarian agents the minimum and maximum score vary markedly. The individual differences are explained by the random influences that are built in through both inclination and grouping. The decision rule appears to be fairly stable to variations in propensity to vote among the act utilitarian agents.

There are stable decision rules that allow act utilitarian agents to function successfully with a minimal requirement of information. The situations in which act utilitarian agents outperform rule utilitarian agents are by no means artificial. The success of act utilitarian agents depends to an important extent on the availability and costs of information, and on the decision rule. Contrary to Harsanyi's claim act utilitarian agents can successfully and spontaneous coordinate their actions. This requires a somewhat different decision rule.

5 Discussion and Conclusions

We have examined Harsanyi's arguments in favor of rule utilitarianism over act utilitarianism. His arguments stand within a heated (old) discussion on which version of utilitarianism is to be preferred. His claim that "...the basic question we need to ask is this: Which version of utilitarianism will maximize social utility? Will society be better of under one or the other? This test very clearly gives the advantage to rule utilitarianism" was tested in an experimental setting. And, not only does Harsanyi makes a particular claim, he also provides the criterion by which the claim is to be judged: maximization of social utility. Since social utility is defined as the basis of the morally good in utilitarianism, our (experimental) approach of calculating social utility scores for different utilitarian rules is in the heart of the philosophical discussion on what is the preferred version.

The experiments executed show that act utilitarian agents need not fare worse than rule utilitarian agents in certain circumstances. This is especially remarkable if one takes into account that they can achieve their results by epistemically less demanding assumptions. They are also able to exploit the surplus of votes when they have the majority to gain some additional utility. This compensates for their occasional loss of the vote due to imperfect (wrong) expectations about the number of fellow act utilitarian that will show up. Core at this ability to perform fairly well is a small relaxation of the decision rule as presented by Harsanyi. It consists of allowing some degree of uncertainty into the decision rule.

The experiments we ran are limited in scope and are open to several objections. Actually, several other assumptions may be chosen with respect to both act utilitarian agents and rule utilitarian agents. We are aware of the fact that the corresponding experiments may yield still other outcomes. We have planned to perform such additional philosophical experiments in the near future. So, at this moment, none of the conclusions and observations we made in this paper are conclusive. But at least we hope to have shown - and that is the main message based on the research performed sofar - that setting up experiments is a useful way to gain new and deeper insights in existing argumentations used in old (and new) philosophical debates.

References

[1] M.E. Bratman, *Intention, Plans and Practical Reasoning*, Harvard University Press, Cambridge, 1981.

[2] P. Danielson, *Artificial Morality*, Routledge, London, 1992.

[3] *Modeling Rationality, Morality and Evolution*, ed., P. Danielson, Oxford University Press, New York, 1998.

[4] J.C. Harsanyi, 'Morality and the Theory of Rational Behaviour', in *Utilitarianism and Beyond*, ed., Sen et al., (1982).

[5] John von Neumann and Oskar Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, NJ, 3rd edn., 1953.

[6] Vincent Wiegel, *SophoLab, Experimental Computational Philosophy*, Ph.D. dissertation, Faculty of Technology, Policy and Management, Delft University of Technology, 2007.

Higher-Order Knowledge in Computer Games

Andreas Witzel and **Jonathan A. Zvesper**[1]

- *Hello, I'm looking for the room for making love.*
- *Oh, right. You must mean the Honeymoon Suite. Well, that's straight that way, can't miss it.*
- *I know where it is. I just wanted you to know that I know where I'm going, so you needn't bother with me.*

"The Missing Ingredient"
First part of the movie "Four Rooms" (1995)

Abstract. Our main aim is to raise awareness of higher-order knowledge, i.e. knowledge about knowledge, as an issue in simulating realistic non-player characters in computer games. We motivate the importance of higher-order knowledge with arguments, as well as a few examples. We survey existing games and literature to show that this issue is currently neglected. We also refer to our earlier work to illustrate one approach to simulating higher-order reasoning, which we call "explicit knowledge programming". Finally we describe a number of issues which arise when carrying out such an implementation, some of which go beyond the scope of the present motivation of computer gaming, and are of more general interest.

1 Introduction

If you ask people why they do things, they often give you reasons in terms of knowledge and beliefs.[2] The answer to the question, "Why did the chicken cross the road?", i.e., "To get to the other side", does not tell the whole story: if we were to attribute human agency to our feathered agent, then a complete answer would be couched in terms of her beliefs and desires. Because we are social animals, these reasons sometimes will involve several agents. In order to reason about interactive behaviour amongst such groups of agents, we have recourse to *higher-order* beliefs (which are beliefs about beliefs). For example, if Ann knows that Bob knows that the party starts at 8, then she usually won't tell him, unless she wants to make sure that he knows that she knows it. Indeed, how could one reason about such a situation without talking about higher-order beliefs? Even if Paul Churchland were correct, and one's talk of beliefs were unscientific nonsense, still it is pervasive and has some explanatory force. If people engage, consciously or unconsciously, in reasoning about beliefs, then a natural way to simulate some aspects of human behaviour, particularly those aspects involving interaction, is to engage in reasoning about beliefs.

[1] ILLC, University of Amsterdam, and CWI, Amsterdam, The Netherlands, e-mail:`{awitzel,jonathan}@illc.uva.nl`

[2] We view "belief" as a more general notion than "knowledge": knowing something implies believing it. Concepts are defined in analogous ways for belief and knowledge, and we will use "epistemic" to mean "of/about beliefs/knowledge". In the more technical parts we will for simplicity focus on knowledge.

An important goal of interactive fiction (IF) games or, more generally, computer role playing games (RPGs) is to simulate convincingly social situations within a virtual world. We believe that to this end epistemic reasoning is crucial. It would therefore be natural for a programmer to be able to describe the behaviour of computer-simulated, non-player characters (NPCs) using rules that contain explicit knowledge statements, including higher-order ones. However, as we will see in Section 2, in current computer games this epistemic aspect is surprisingly overlooked.[3] To strengthen our argument, in Section 3 we will give some scenarios, involving higher-order epistemic reasoning, that we find plausible to occur in RPGs.

In [28] we described an approach for providing knowledge statements on the level of a programming language, and for proving that such statements are evaluated faithfully with respect to a formal definition of knowledge. That approach is modular in the sense that all aspects of epistemic modeling can be implemented in a dedicated knowledge module. In the context of computer games, an NPC scripter could thus use high-level tools and would be alleviated from having to keep track of these things explicitly and manually. We will briefly describe that approach, which uses ideas from epistemic logic, in Section 4.

We will conclude by raising some open issues in Section 5. These are concerned with making the knowledge module tractably implementable by restricting the possible inputs and queries. The basic questions are:

- What kinds of events should be used as inputs to change the state of the module, and how to formalize them? Even a simple event, like the player picking up an object, may have an effect on some agents' (higher-order) knowledge. These effects will depend on subtleties of the wider situation.
- What kinds of queries to the module are relevant? Not all theoretically possible knowledge statements will occur or even be realistic to compute. Since the goal is to simulate real-life social interaction, to answer this question one has to look at human higher-order reasoning.

2 State of the Art

We will briefly review the current state of the art with respect to epistemic modeling in computer games, both in the "real world" and in (academic) research.

2.1 Real World

The state of the art in commercial computer games is not easy to judge, since computer game companies are not very keen on publish-

[3] When we refer to "computer games" in this paper, we have games like RPGs and IF in mind. For other games, like poker, epistemic aspects have indeed been considered.

ing the details of their AI implementations. So if one doesn't want to rely on the enthusiastic slogans from marketing departments, then the best sources of information about computer game AI are private web pages like [29] where observations and analyses from playing, interview quotations, and possibly the website creator's own knowledge and experience as AI programmer are carefully collected and presented. For an extensive list of online resources, see [23].

From these resources it becomes evident that epistemic reasoning is definitely not in the focus of existing computer game AI, and we did not find any mention of higher-order reasoning. For example, the highly acclaimed Radiant AI engine is used in the RPG *The Elder Scrolls: Oblivion*TM [6] to make the game more lifelike. The following quotation from an interview [5] during the testing phase of the game AI illustrates its functioning:

> One [non-player] character was given [by the testers] a rake and the goal "rake leaves"; another was given a broom and the goal "sweep paths," and this worked smoothly. Then they swapped the items, so that the raker was given a broom and the sweeper was given the rake. In the end, one of them killed the other so he could get the proper item.

Obviously, the characters didn't mutually know their interests, or they couldn't make use of that knowledge.

To us it seems natural that one would use a logic-based approach in order to effectuate epistemic reasoning. Indeed, a logic-based formalism is an important part of what we will suggest in Section 4.

References in these directions are scarce. In [18], it is suggested to use logic for NPC scripting; however, higher-order epistemic reasoning is not considered, and the article seems to have been left at a brainstorming stage and not followed up on. The clearest statement promoting the use of epistemic reasoning comes from the famous IF writer Emily Short [25]:

> *Abstract Knowledge.* One of the artificial abilities we might like to give our NPCs, aside from the ability to wander around a map intelligently and carry out complex goals, is the ability to understand what they are told: to keep track of what items of knowledge they have so far, use them to change their plans and goals, and even draw logical inferences from what they've learned.

It is not clear whether this refers to higher-order knowledge, or whether "abstract" just is meant to imply that the implementation should be generic and encapsulated in a knowledge module; in any case, the currently existing IF implementation of such ideas [11] is restricted to pure facts and does not include any reference to the possibility of higher-order knowledge.

2.2 Research

Computer game AI is slowly becoming accepted as an area for serious academical research [15, 13]. In recent years, conferences and other scientific meetings on the interface of artificial intelligence and computer games have emerged [24, 4, 17] and special issues of magazines have appeared [1, 2, 3].

Again, where knowledge is considered, the concern seems to be exclusively domain knowledge, or knowledge about facts in the game world, as in [22, 26]. A more general approach of using agent programming languages to script NPCs (e.g. [16]) inherits the epistemic reasoning facilities of such languages – which tend to focus on facts. The closest in spirit to higher-order modeling are attempts to detect the general attitude of the human player (for example, aggressive or cautious) and to adjust the game AI accordingly. But we could find no references to explicit higher-order epistemic modeling.

The ScriptEase system [8] is an academic approach to NPC scripting, which was motivated by the insight that the scripting process needs to be simplified. It provides a graphical interface for generating behaviours of characters in a commercial RPG. However, knowledge statements to steer the behaviour are not considered.

A very interesting approach, described in [9], uses deontic logic to specify NPC behaviour in a rule-based fashion. While epistemic issues are not considered there, a fusion of these two aspects could provide a highly suitable system for scripting believable social agents.

Some literature on higher-order reasoning in multi-agent systems that does *not* focus on computer games is also very relevant. In [10], the specific problem of agent *communication* is studied, in which agents weigh costs against expected benefit of communication. The authors point out the importance of using higher-order reasoning, in the form of beliefs about beliefs, when agents make such assessments. Their particular interest is in formal representation of belief "*abduction*". We do not consider abductive reasoning here, but we recognise that it is also important in our settings.

We also note that higher-order reasoning is discussed in [30] in the context of a Petri Net method for designing "intelligent team training systems". The authors suggest that using Petri Nets can help to overcome tractability issues in epistemic reasoning. However, they note that communication, an important ingredient in the kind of social interaction we wish to simulate, "is more complicated than Petri Nets can represent". We do not consider the Petri Net formalism further, but if progress is made in this area it could be of relevance.

3 Scenarios and Examples

Having seen that higher-order epistemic reasoning is not currently considered in the context we are interested in, we will now describe a few small scenarios of social interaction naturally involving this kind of reasoning. We believe that by simulating the aspects of the real world that are highlighted by these examples, computer game worlds will become more convincing.

Scenario 1. If Ann is (openly) present when Carl tells Bob about some fact, then she won't tell the same fact to Bob again a minute later. Indeed, doing so under usual circumstances would be puzzling and inexplicable behaviour.

Scenario 2. If Ann gets an advantage from lying to Bob, and knows that he doesn't know the truth then she might indeed lie; if she knows he does know then she usually won't; if she doesn't know whether he knows then her decision may depend on other circumstances.

Scenario 3. Part of being a doctor is the obligation to help people around you whom you know to be sick.[4] Imagine Ann is a doctor who unfortunately would profit of getting rid of Bob. Not only will she take care that no-one sees her putting the poison into his glass, she will also make sure (e.g. by immediately going on holidays) that no-one knows that she knows that Bob is dying, because otherwise not saving him would make her suspicious.

The simple rule of pragmatics in Scenario 1 (that things which are commonly known[5] aren't usually worth stating) makes essential use

[4] This scenario is inspired by a discussion from [20].
[5] Something is common knowledge among a group of agents if all agents know it, know that they all know it, and so on ad infinitum.

of epistemic reasoning. Moreover, whatever character type Ann in Scenario 2 is supposed to have, implementing it should be facilitated by epistemic operators. The last scenario may seem a bit contrived, but on the other hand it is very common for game characters to have professions, and in a world of intrigue and adventure having to get rid of someone is also quite conceivable. It manifestly also involves epistemic reasoning.

The variety and generality of these scenarios illustrates how applicable the underlying ideas are in many different situations which could occur in IF or RPGs.

4 Making Knowledge Explicit

If more attention should be paid to epistemic reasoning in simulating human interaction in IF and RPGs, a natural question is: how should this be done? In this section we propose an approach for tackling the problem of programming a simulation of higher-order epistemic reasoning.

The approach is based around epistemic logic.[6] That is, we propose using some formal language in which epistemic statements can be formulated and evaluated. The formulae of such a language might for example be built recursively from "atoms", which are non-epistemic facts, by using propositional connectives, like $\vee$ (or) and $\neg$ (not), and knowledge operators K_a, one for each agent a being simulated. We will call these "epistemic formulae". So for example if p were an atom, then K_aK_bp could be an epistemic formula with the intended reading "a knows that b knows that p".

In [28], we described a simple and preliminary implementation to provide statements involving explicit knowledge formulae on the programming language level, and proved the implementation to be sound with respect to a formal notion of knowledge defined on the level of the underlying process calculus. We took a modular approach, writing a knowledge module that is instantiated for each process. We will briefly review this work in the following.

In our particular implementation, the events that were used as inputs to the knowledge module were always synchronous communications between two of the processes concerning the values of some bits. In general though, an event can be anything which would have epistemic effects. The idea is to pass to the knowledge module of a process a the events that a 'observes'.

The queries to which the knowledge module can respond are epistemic formulae. We used a formal language with atoms $p_{x_0}, \neg p_{x_0}, p_{x_1}, \neg p_{x_1}, \ldots$ for each of the bits $x_0, x_1, \ldots$, and a knowledge operator $K_a, K_b, \ldots$ for each of the processes $a, b, \ldots$.[7] Then, as an example, the formula $K_cK_b\neg p_{x_2}$ means that process c knows that process b knows that x_2 has the value 0. If the knowledge module of process a were passed this formula it could respond by saying "yes", "no" or "don't know". If the module were to respond "yes", then this should be interpreted as (the agent which is modelled by process) a knowing that c knows that a knows that x_2 is 0. A programmer can use such queries, for example, in conditional statements and thus have the program flow depend on the process's knowledge.

Even with a simple implementation, it was desirable to prove that it was in some sense "correct". Thus we used a formalism from the literature on epistemic logic, namely Kripke models. The argument for correctness of the implementation then proceeds in two steps, which can roughly be stated as follows:

- Argue that a particular model $\mathcal{M}$ represents faithfully the intuitive situation which we intended to capture.
- Prove that knowledge formulae are evaluated in the same way by process a after the sequence of events σ as they are by agent a in the model $\mathcal{M}$ after the same sequence of events.

One criticism that one can make of using a Kripke model formalism as an intermediate step is that that formalism itself can appear unintuitive. However, we know of no more philosophically grounded and mathematically robust formalism with which to work in the context of reasoning about higher-order knowledge. (In order to deal with various phenomena like so-called "explicit belief", or inconsistent beliefs, many other models have been proposed, but these are all essentially refinements or variations of Kripke models – see [19, Sections 2.4 to 2.13] for a selective survey.)

The Kripke model that we specified for the particular implementation we had in mind resembled an "interpreted system" from [12]. It was not our aim to implement directly an entire interpreted system, which in this case is an infinite structure. Even if it is finitely representable, we might only be interested in certain parts of it.

In general, an implementation can be simplified by considering a subclass of the formulae that would be in the full logical language which the model could interpret. As it happened, for the particular implementation we had in mind (a distributed implementation of an algorithm for eliminating dominated strategies in strategic games), it was only necessary to consider formulae from the very simple epistemic language that we have described.

In the case of simulating realistic *human* agents (so in particular within IF and RPGs), the limits to human cognitive faculties should be taken into account. So for example, would it make sense to allow as queries to the knowledge module epistemic formulae involving complex iterations as in, "Ann believes that Bob believes that Carl doesn't believe that Ann believes that Derek believes that it's raining"?

5 Open Issues

This brings us to the question of what inputs and queries to the knowledge module ought to be allowed. The two parts of this question are to some extent independent.

The first part concerns the events in the game world that should affect the knowledge states of the characters. In a way the issue of what events should be taken into account is up to the designer of the world. We think it is difficult to make a general statement about which kinds of events matter and which don't.

Once the events that the virtual world generates are more or less decided, we need abstract representations of them and specifications of their exact conditions and epistemic effects. Again it is difficult to make general statements, because they depend on the specific event. For example, if an event consists in the player picking up an object, the simplest approach would be that it becomes common knowledge between all present agents that the player possesses that object. But there is some freedom in the degree of fine-grainedness and detail in which this event should be represented and processed. Should simply all agents in a certain radius gain common knowledge, or should it be taken into account in which direction they are currently looking and whether everyone is mutually visible?

The second part of the question, which is clearer than the previous part and which we believe may be answered empirically, concerns

[6] The research field of epistemic logic can be said to have been initiated with [14].

[7] Note that here we are not using the richer language that could be built using also the connectives $\neg$ and $\vee$ mentioned above.

the class of knowledge statements that matter and that the knowledge module should be able to handle. Here again the fact can be used that the computer games we have in mind want to realistically model human social interaction. The question, which is also of independent interest, then becomes: What class of epistemic formulae are evaluated by humans in everyday life, consciously or not?

Some results from experimental game theory about levels of strategic thinking [7] can be interpreted as being relevant to this question. However, these experiments do not focus on everyday social interaction. For example, the Beauty Contest game mentioned in the previous reference might invoke conscious and explicit reasoning about the other agents, while we believe that, through years of experience in social interaction, the requisite higher-order reasoning processes may also occur on a more intuitive and reflexive level.

Furthermore, the experimental designs are in general not specifically concerned with knowledge, so that at best the results can give us hints about the nesting depth of knowledge operators. Clearly other criteria might define the class of knowledge formulae that are of relevance in social interactions.

For example, from Scenario 2 it is clear that formulae like

$$K_a \neg K_b p$$
$$K_a K_b p$$
$$\neg(K_a K_b p \vee K_a \neg K_b p)$$

matter. However, that doesn't necessarily hold for all formulae of knowledge operator depth 2. Also, human reasoning capabilities may not be monotonic with respect to this complexity measure. For example, for a special concept like common knowledge, which in theory involves infinite depth, we might want to assume that humans are able to cope with it, while this does not hold for "intermediate" depths like 10000.

The main issue thus remains: How can we define this class of relevant formulae?

In [27], results from experiments suggest that subjects use first-order Theory of Mind (beliefs about others' beliefs), but not "all kinds of reasoning possible and useful in this context". This supports the claim that the depth of knowledge operators is not the only relevant criterion. It is further reported that some subjects use second-order Theory of Mind, which corresponds to third-order epistemic formulas.

While it is interesting to look at such questions in experimental setups, an alternative approach could shed additional light on these issues, namely to come up with realistic social situations and think about what kinds of reasoning processes go on. This is basically what we did in Section 3, and work by Parikh is an excellent source for enlightening examples (see e.g. [21]). Such thought experiments or observations from real life can be convincing enough to remove the need for abstractions and reproducible lab conditions as provided by experiments, for the benefit of being set in more natural environments, where human reasoning capabilities possibly profit from experience and training in specific social situations.

6 Summary

The main aim of this paper was to raise awareness of higher-order knowledge as an issue in simulating realistic non-player characters in computer games. Section 2 surveyed existing games and literature and found that this has not yet been done. Section 3 gave some illustrative and motivating examples of what we mean by higher-order knowledge, in situations that could plausibly occur in IF and RPGs. Section 4 gave an example of an implementation of explicit knowledge programming. One suggestion of this paper is that this is a sensible and realistic approach to implementing higher-order knowledge reasoning, and therefore to simulating some interactive aspects of human behaviour. Finally, Section 5 described a number of issues which arise when carrying out such an implementation. We raised issues which go beyond the scope of the present motivation of IF and RPGs, and are of general interest.

Acknowledgments

We would like to thank three anonymous reviewers for their very helpful comments.

This research was supported by a Marie Curie Early Stage Research fellowship in the project GLoRiClass (MEST-CT-2005-020841).

REFERENCES

[1] *IEEE Intelligent Systems*, **17**(4), (Jul/Aug 2002). Special Issue on Interactive Entertainment.

[2] *IEEE Intelligent Systems*, **21**(5), (Sept/Oct 2006). Special Issue on Interactive Entertainment.

[3] *Science of Computer Programming*, **67**(1), (June 2007). Special Issue on Aspects of Game Programming.

[4] *Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, eds., David W. Aha, Héctor Muñoz-Avila, and Michael van Lent, 2005.

[5] Wikipedia article. The Elder Scrolls IV: Oblivion. `http://en.wikipedia.org/w/index.php?title=The_Elder_Scrolls_IV:_Oblivion&oldid=96078473#Radiant_A.I.`, December 2006. (The relevant section was removed in later revisions, since it concerned "Radian AI's behavior prior to the game release. Listing all these examples is beyond the scope of this article." We did not succeed to find the original interview from which the section was taken.).

[6] Bethesda Softworks. The Elder Scrolls: Oblivion. `http://www.elderscrolls.com/games/oblivion_overview.htm`, 2006.

[7] Colin F. Camerer, 'Behavioural studies of strategic thinking in games', *Trends in Cognitive Sciences*, **7**, 225–231, (May 2003).

[8] Maria Cutumisu, Curtis Onuczko, Matthew McNaughton, Thomas Roy, Jonathan Schaeffer, Allan Schumacher, Jeff Siegel, Duane Szafron, Kevin Waugh, Mike Carbonaro, Harvey Duff, and Stephanie Gillis, 'Scriptease: A generative/adaptive programming paradigm for game scripting', *Science of Computer Programming*, **67**, 32–58, (June 2007).

[9] Flávio S. Corrêa da Silva and Wamberto W. Vasconcelos, 'Rule schemata for game artificial intelligence', in *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, Ribeirão Preto, Brazil, (2006). Springer.

[10] Aldo Franco Dragoni, Paolo Giorgini, and Luciano Serafini, 'Mental states recognition from communication', *Journal of Logic and Computation*, **12**, 119–136, (February 2002).

[11] Eric Eve. Epistemology (Inform 7 Extension). `http://www.inform-fiction.org/I7Downloads/Extensions/Eric%20Eve/Epistemology`.

[12] Ronald Fagin, Joseph Y. Halpern, Moshe Y. Vardi, and Yoram Moses, *Reasoning about knowledge*, MIT Press, 1995.

[13] Chris Fairclough, Michael Fagan, Brian Mac Namee, and Padraig Cunningham, 'Research directions for AI in computer games', in *Twelfth Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 333–344, (2001).

[14] Jaakko Hintikka, *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell, 1962.

[15] John E. Laird and Michael van Lent, 'Human-level AI's killer application: Interactive computer games', *AI Magazine*, 15–25, (2001).

[16] João Leite and Luís Soares, 'Evolving characters in role playing games', in *18th European Meeting on Cybernetics and Systems Research (EMCSR 2006)*, volume 2, pp. 515–520, Vienna, (2006).

[17] Sushil J. Louis and Graham Kendall, eds. *Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games (CIG06), University of Nevada, Reno, campus in Reno/Lake Tahoe, 22-24 May, 2006*. IEEE, 2006.

[18] Sami Mäkelä. NPC Scripting and Reasoning about the NPC behaviour. `http://www.worldforge.org/project/newsletters/November2001/NPC_Scripting`.

[19] John-Jules Ch. Meyer and Wiebe van der Hoek, *Epistemic Logic for AI and Computer Science*, Cambridge University Press, 1995.

[20] Eric Pacuit, Rohit Parikh, and Eva Cogan, 'The logic of knowledge based obligation', *Synthese*, **149**(2), 311–341, (March 2006).

[21] Rohit Parikh, 'Levels of knowledge, games, and group action', *Research in Economics*, **57**, 267–281, (September 2003).

[22] Marc Ponsen, Pieter Spronck, Héctor Muñoz-Avila, and David W. Aha, 'Knowledge acquisition for adaptive game AI', *Science of Computer Programming*, **67**, 59–75, (June 2007).

[23] Craig Reynolds. Game Research and Technology. `http://www.red3d.com/cwr/games/`.

[24] Jonathan Schaeffer and Michael Mateas, eds. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, June 6-8, 2007, Stanford, California, USA*. The AAAI Press, 2007.

[25] Emily Short. Conversation. `http://emshort.wordpress.com/writing-if/my-articles/conversation`.

[26] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma, 'Adaptive game AI with dynamic scripting', *Machine Learning*, **63**, 217–248, (June 2006).

[27] Rineke Verbrugge and Lisette Mol, 'Learning to apply theory of mind', *Journal of Logic, Language, and Information*, (2008). To appear.

[28] Andreas Witzel and Jonathan Zvesper, 'Epistemic logic and explicit knowledge in distributed programming (short paper)', in *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, (2008). To appear.

[29] Steve M. Woodcock. The Game AI Page. `http://www.gameai.com`.

[30] Jianwen Yin, Michael S. Miller, Thomas R. Ioerger, John Yen, and Richard A. Volz, 'A knowledge-based approach for designing intelligent team training systems', in *Agents*, pp. 427–434, (2000).