

# VaToMAS – Verification and Testing of Multi-Agent Systems

Edited by

Alessio R. Lomuscio<sup>1</sup>, Sophie Pinchinat<sup>2</sup>, and Holger Schlingloff<sup>3</sup>

1 Imperial College London, GB, [A.Lomuscio@imperial.ac.uk](mailto:A.Lomuscio@imperial.ac.uk)

2 IRISA – Rennes, FR, [sophie.pinchinat@irisa.fr](mailto:sophie.pinchinat@irisa.fr)

3 HU Berlin, DE, [hs@informatik.hu-berlin.de](mailto:hs@informatik.hu-berlin.de)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13181 “VaToMAS – Verification and Testing of Multi-Agent Systems”.

**Seminar** 28. April–03. May, 2013 – [www.dagstuhl.de/13181](http://www.dagstuhl.de/13181)

**1998 ACM Subject Classification** I.2.11 Distributed Artificial Intelligence, F.3.1 Specifying and Verifying and Reasoning about Programs, I.2.4 Knowledge Representation Formalisms and Methods

**Keywords and phrases** Model checking, Specification-based testing, Multi-agent systems, Controller synthesis, Temporal logic

**Digital Object Identifier** 10.4230/DagRep.3.5.151

**Edited in cooperation with** Bastien Maubert

## 1 Executive Summary

*Alessio R. Lomuscio*

*Sophie Pinchinat*

*Holger Schlingloff*

**License**  Creative Commons BY 3.0 Unported license  
© Alessio R. Lomuscio, Sophie Pinchinat, and Holger Schlingloff

Multi-agent systems (MAS) are distributed computing systems in which the individual components, or agents, interact with each other by means of communication, negotiation, cooperation etc., in order to meet private and common goals. The agent model finds applications in a variety of key applications of high-impact to society including web-services, autonomous vehicles, and e-government. But if MAS are to deliver on their promise to drive future applications, they need to be reliable.

MAS are typically specified and reasoned about by a variety of modal formalisms, including a variety of different logics. There are presently several, compartmented communities tackling questions pertaining to the correctness of MAS: researchers in model checking, model based testing, and controller synthesis. There presently is very little personal interaction among the scientists from different communities. The aim of this seminar was to bring these communities together, get exposure to each others’ solutions to similar aims, and ultimately enhance their future interaction.

The topics concentrated on the intersection of the fields:

- Model checking of temporal-epistemic logic, alternating logics, and BDI logics
- Model based test generation for embedded systems
- Controller synthesis for self-organizing systems



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

VaToMAS – Verification and Testing of Multi-Agent Systems, *Dagstuhl Reports*, Vol. 3, Issue 4, pp. 151–187

Editors: Alessio R. Lomuscio, Sophie Pinchinat, and Holger Schlingloff



DAGSTUHL  
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In model checking, usually a model of the system and a property to be verified are given. In model based test generation, the goal is to construct a test suite from a model which establishes confidence in a certain property. In synthesis, a property and a model of computation are given, from which a strategy (a system model) is to be built. Both the test generation and the controller synthesis problem are closely related to model checking – in order to check the satisfiability of certain alternating time temporal logic (ATL) formulas in a model, one needs to construct a strategy for the participating agents.

The purpose of the seminar was to establish a common understanding of the problems in the different technologies of these application areas. It was expected that increased interaction between these three fields would stimulate new results and techniques of both theoretical relevance and practical usefulness.

Besides survey talks (60 minutes) on common technologies, attendees gave short contributions (30 minutes) and lightening presentations (15 minutes) on current research results and discussion rounds on open problems and research agendas. Additional technical sessions, including software demos, were organised spontaneously by the attendees for two of the evenings.

Attendees also contributed to the seminar by taking part in the lively discussions organised on topics of importance in the area. These were held in some of the afternoons but also at during informal occasions outside the usual seminar hours such as after dinner. This helped bridge some of the gaps between the subdisciplines and rectify some misconception about each other's work.

Specifically, research topics of the seminar included:

- Logics and specification formalisms for MAS
- Verification and model checking for interactive systems
- Model-based testing for MAS
- Explicit, symbolic, and SAT-based algorithms for module checking
- Test case generation and synthesis
- Synthesis of winning strategies for games

The goals of the seminar were

- to obtain a common understanding of base technologies and intersections between these topics
- to collect a state-of-the-art picture of recent research results in the fields
- to confront methods from model checking and test generation for MAS
- to clarify terminology, research agendas and open problems
- to define a set of benchmark problems for verification and testing of MAS
- to bring together different communities formerly not interacting

The research topics were also discussed in relation with embedded systems applications such as:

- Verification of cyber-physical systems
- Validation of autonomous robots

It was felt that the seminar helped the participants to reach a common and shared understanding on the roles of logic, verification and testing as well as their interplay in the context of multi-agent systems

## 2 Table of Contents

### Executive Summary

*Alessio R. Lomuscio, Sophie Pinchinat, and Holger Schlingloff* . . . . . 151

### Overview of Talks

The Social Laws Paradigm for Coordinating Multi-Agent Systems <i>Thomas Ågotnes</i> . . . . .	156
(Really) Dynamic Modal Logics <i>Carlos Areces</i> . . . . .	156
From Control Theory to Game Theory via LTLKc <i>Guillaume Aucher</i> . . . . .	157
On Decentralized Runtime Verification Techniques <i>Ezio Bartocci</i> . . . . .	157
Automatic Verification of Equational Knowledge in MAS <i>Ioana Boureanu</i> . . . . .	158
Protocol Descriptions to Interpreted Systems <i>Ioana Boureanu</i> . . . . .	158
Alternating Epistemic Mu-Calculus: Fixed-point Abilities under Incomplete Information <i>Nils Bulling</i> . . . . .	159
Combining quantitative and qualitative strategic reasoning. Part II: some comparisons and preliminary results <i>Nils Bulling</i> . . . . .	159
Using AJPF to generate models of agent programs for input into other Model Checkers <i>Louise A. Dennis</i> . . . . .	160
Verifying Autonomous Systems <i>Michael Fisher</i> . . . . .	160
Resolution for Temporal Logics of Knowledge <i>Michael Fisher</i> . . . . .	160
Information values in multi-agent bargaining scenarios <i>Tim French</i> . . . . .	161
The synthesis and actuation of informative events <i>Tim French</i> . . . . .	161
Combining quantitative and qualitative strategic reasoning. Part I: framework <i>Valentin Goranko</i> . . . . .	161
A few remarks about related work in Pretoria <i>Stefan Gruner</i> . . . . .	162
Yet Another Modal Notation for Strategy Contexts <i>Dimitar Guelev</i> . . . . .	162
The grand game of testing <i>Yuri Gurevich</i> . . . . .	163

Managing Policies and Trust	
<i>Yuri Gurevich</i> . . . . .	163
Logics for Multi-Agent Systems	
<i>Andreas Herzig</i> . . . . .	164
Concepts, Agents, Strategies... and Coalitions. ATL goes (monadic) first order	
<i>Wojtek Jamroga</i> . . . . .	164
ATL with strategy contexts – part 1	
<i>François Laroussinie</i> . . . . .	164
ATL with strategy contexts – part 2	
<i>Nicolas Markey</i> . . . . .	165
Uniform Strategies	
<i>Bastien Maubert</i> . . . . .	165
A Poor Man’s Technique for Reasoning About Knowledge	
<i>Stephan Merz</i> . . . . .	165
Reasoning About Strategy	
<i>Aniello Murano</i> . . . . .	166
Bounded model checking for LTLK	
<i>Wojciech Penczek</i> . . . . .	167
Abstract planning using genetic algorithms	
<i>Wojciech Penczek</i> . . . . .	167
Tools for MAS verification: where do we go next?	
<i>Franco Raimondi</i> . . . . .	167
Doomsday Equilibria for Omega-Regular Games	
<i>Jean-François Raskin</i> . . . . .	168
Specification based testing in an institutional setting	
<i>Markus Roggenbach</i> . . . . .	168
Efficient Testing of Software Product Lines	
<i>Ina Schaefer</i> . . . . .	169
On the specification and analysis of contracts (normative texts)	
<i>Gerardo Schneider</i> . . . . .	169
Flatland logic	
<i>François Schwarzentruber</i> . . . . .	169
Before the announcement	
<i>Hans van Ditmarsch</i> . . . . .	170
Scaling up Test Data Generation	
<i>Ramanathan Venkatesh</i> . . . . .	171
Show me your friends and I tell you who you are	
<i>Karsten Wolf</i> . . . . .	171
Synthesis of Knowledge-Based Program Implementations	
<i>Ron van der Meyden</i> . . . . .	171


**Working Groups**

Case Study Description: Elevators <i>Working Group 2</i> . . . . .	172
Case Study Description: Access Control <i>Dimitar P. Guelev</i> . . . . .	174
Case Study Description: Earthquake Rescue Mission <i>Working Group 1</i> . . . . .	175
Case Study Description: Examples from dialog/social systems <i>Working Groups 3 and 4</i> . . . . .	176
Case Study Description: Avionic scenario <i>Franco Raimondi</i> . . . . .	180
Discussion about the testing of MAS <i>Working Group 7</i> . . . . .	185
Discussion concerning logics for knowledge, time and strategies <i>Working Group 5</i> . . . . .	186
<b>Participants</b> . . . . .	187

## 3 Overview of Talks

### 3.1 The Social Laws Paradigm for Coordinating Multi-Agent Systems

*Thomas Ågotnes (University of Bergen, NO)*

License  Creative Commons BY 3.0 Unported license

© Thomas Ågotnes

Joint work of Ågotnes, Thomas; van der Hoek, Wiebe; Wooldridge, Michael

Social laws (or normative systems) have emerged as a natural and powerful paradigm for coordinating such systems, exposing the whole spectrum between fully centralised and fully decentralised coordination mechanisms. A social law is, intuitively, a constraint on the behaviour of agents, which ensures that their individual behaviours are compatible. In a standard multi-agent state transition diagram (where transitions are labelled with the name of the agent effecting the transition), a social law is simply a labelling of the transitions saying whether or not they are “legal”, or “desirable”. An illegal transition could, for example, correspond to a component malfunctioning. Different social laws give rise to different global system properties, assuming that the social law is complied with. Such properties can be specified and verified using temporal modal logic and model checking. In the talk I will introduce and motivate the idea of social laws, and show how many key social laws verification problems can be solved using model checking. Key questions involve the notion of compliance. First, when is it rational for an agent to comply? Since the properties of the resulting system depend on compliance of all the agents in the system (think of the social law “drive on the right side of the road”), this requires a game theoretic analysis. Second, how can we identify the most important agents/components in the system, the agents whose compliance is crucial for the proper functioning of the system? I will talk about some resulting decision problems, combining logic, game theory, voting theory and complexity theory.

### 3.2 (Really) Dynamic Modal Logics

*Carlos Areces (Universidad Nacional de Cordoba, AR)*

License  Creative Commons BY 3.0 Unported license

© Carlos Areces

Different Dynamic Modal Logics have been investigated in the literature (e.g., Propositional Dynamic Logics). Interestingly, the semantics of most of these logics is actually *static*: the model over which a formula is evaluated never changes. We are interested in logics with operators that can actually alter the model during evaluation. We will present a number of these operators, discuss their expressive power, and the complexity of the model checking and satisfiability problems.

### 3.3 From Control Theory to Game Theory via LTLKc

*Guillaume Aucher (INRIA Bretagne Atlantique – Rennes, FR)*

License © Creative Commons BY 3.0 Unported license  
© Guillaume Aucher

Supervisory control theory as initiated by Ramadge and Wonham has been developed independently from game theory. In this talk, we show that it is possible to embed infinite two-player games with imperfect information and safety objective into an extension of supervisory control theory with infinite words. In order to do so, we use an epistemic temporal logic as a lingua franca, and we reformulate the problems of supervisory control theory and infinite games with imperfect information into this logic. As a result of this embedding, we are able to compute and characterize completely in terms of model checking problems the set of winning strategies in a two-player game with imperfect information and safety objectives.

This talk is based on a paper available as a technical report at the HAL archive <http://hal.inria.fr/>.

### 3.4 On Decentralized Runtime Verification Techniques


*Ezio Bartocci (TU Wien, AT)*

License © Creative Commons BY 3.0 Unported license  
© Ezio Bartocci

Most of computing devices produced nowadays are embedded systems employed to monitor and control physical processes: cars, airplanes, automotive highway systems, air traffic management, etc. In all these scenarios, computing and communicating devices, sensors monitoring the physical processes and the actuators controlling the physical substratum are distributed and interconnected together in dedicated networks. The formal verification of these systems consists in proving that their execution satisfies a given specification of what their possible behaviors should be. When a system model is available and has a finite number of states, an answer is provided by applying the classical Model Checking technique. In this case, if the system is found to violate the property, a counterexample in the form of a system execution is generated and it can be used to debug the system model or property. The main drawback of this technique is that usually the number of states of a model grows exponentially in the number of its parameters. Furthermore, often the models are not available, leading us to consider them simply as black-boxes where only input and output can be observed. In this case, monitoring their execution provides a still valid alternative verification technique. Decentralized runtime verification refers to a monitoring technique, where each component must infer, based on a set of partial observations, if the global property is satisfied. In this talk I will present an overview of the problem, the available techniques, some of my research results and points of discussion.

### 3.5 Automatic Verification of Equational Knowledge in MAS


*Ioana Boureanu (EPFL – Lausanne, CH)*

License  Creative Commons BY 3.0 Unported license  
© Ioana Boureanu

Security protocols can be executed concurrently, their participants and their runs describing a MAS. However, as we may know, the knowledge of these agents is subject to the cryptographic abilities and, roughly speaking, to the cryptographic material that they hold. We would like to be able to describe this knowledge in a formal way. Per se, this is not novel. The element of originality is two-fold: 1) being able to treat many primitives, i.e., beyond simple encryption/decryption and, namely, the full class of subterm convergent equational, cryptographic theories; 2) finding a solution that lends itself to automation and then to unsupervised verification. We introduce a modality called rewriting knowledge that operates on local equational congruences. We discuss the conditions under which its interpretation can be approximated by a second modality called empirical knowledge. We report an implementation of a technique to verify this modality inside the open source model checker MCMAS. We evaluate the approach by verifying MAS models of electronic voting protocols automatically extracted from high-level descriptions.

### 3.6 Protocol Descriptions to Interpreted Systems

*Ioana Boureanu (EPFL – Lausanne, CH)*

License  Creative Commons BY 3.0 Unported license  
© Ioana Boureanu

**Main reference** I. Boureanu, A. Lomuscio, “PD2IS Tool”. May, 2013.

**URL** <http://www.dagstuhl.de/mat/Files/13/13181/13181.BoureanuIoana2.ExtAbstract.pdf>

PD2IS (Protocol Descriptions to Interpreted Systems) is a toolkit that we developed to generate MAS models upon standard security protocol semantics, e.g., we embedded the multiset rewriting semantics into IS models. The input to PD2IS is a file designating a CAPSL (Common Authentication Protocol Specification Language) protocol description together with some additional parameters to define a MAS instantiation. The output is an interpreted system in ISPL (Interpreted Systems Programming Language). PD2IS systematically generates a full taxonomy of propositions and temporal-epistemic formulae corresponding to expressions of the CAPSL goals. PD2IS automatically inserts these temporal-epistemic formulae in the ISPL file for the model under generation. MCMAS is called for each ISPL file produced by PD2IS. MCMAS returns the calls either by certifying that the specifications are satisfied or by returning detailed counterexamples. These are used by PD2IS to report details of the attack found on the protocol (i.e., the failure of one or more of formulae corresponding to the goals). PD2IS was used together with MCMAS and proved effective in verifying authentication, key-establishment (e.g., the Clark-Jacobs, SPORE libraries), e-voting protocols (FOO’92, Okamoto protocols), against classical security specifications (secrecy, authentication, vote-privacy, receipt-freeness, coercion-resistance, etc.), as well as novel, intrinsically epistemic security requirements, like attack-detectability.



### 3.7 Alternating Epistemic Mu-Calculus: Fixed-point Abilities under Incomplete Information

Nils Bulling (TU Clausthal, DE)

**License** © Creative Commons BY 3.0 Unported license  
© Nils Bulling

**Main reference** N. Bulling, W. Jamroga, “Alternating epistemic mu-calculus,” in Proc. of the 22nd Int’l Joint Conf. on Artificial Intelligence (IJCAI’11), pages 109–114, Barcelona, Spain, July 2011.

**URL** <http://ijcai.org/papers11/Papers/IJCAI11-030.pdf>

The alternating-time temporal logic ATL is a well-known logic for reasoning about strategic abilities of agents. An important feature that distinguishes the variants of ATL for imperfect information scenarios is that the standard fixed-point characterizations of temporal modalities do not hold anymore. In this talk, I present the alternating epistemic mu-calculus [1]. The logic allows to capture abilities that could not be expressed in ATL. The new kind of ability allows agents to always recompute their strategy while executing it. Thus, the agents are not assumed to remember their strategy by definition. I will also briefly address the model checking problem and show that the verification of such abilities can be cheaper than for all the variants of “ATL with imperfect information” considered so far.

#### References

- 1 Nils Bulling and Wojciech Jamroga. Alternating epistemic mu-calculus. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 109–114, Barcelona, Spain, July 2011.

### 3.8 Combining quantitative and qualitative strategic reasoning. Part II: some comparisons and preliminary results

Nils Bulling (TU Clausthal, DE)

**License** © Creative Commons BY 3.0 Unported license  
© Nils Bulling

**Joint work of** Bulling, Nils; Goranko, Valentin

**Main reference** N. Bulling, V. Goranko, “How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract),” in Proc. of the 1st Int’l Workshop on Strategic Reasoning, EPTCS, Vol. 112, pp. 33–41, 2013.

**URL** <http://dx.doi.org/10.4204/EPTCS.112.8>


In this talk I take our framework on quantitative and qualitative reasoning [1] (presented in Part I) up and propose the logic Quantitative ATL\*. I present some preliminary model checking results and briefly discuss related work.

#### References

- 1 Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *Proceedings of the 1st International Workshop on Strategic Reasoning*, Electronic Proceedings in Theoretical Computer Science, pages 33–41, Rome, Italy, March 2013.

### 3.9 Using AJPF to generate models of agent programs for input into other Model Checkers


*Louise A. Dennis (University of Liverpool, GB)*

License  Creative Commons BY 3.0 Unported license  
© Louise A. Dennis

AJPF is a program model checker for reasoning about agent systems programmed in BDI style languages. Following recent work from Raimondi et al., this has been adapted so that it can be used to generate a model of the system under test, and that model then used as the input for a more traditional model-checker. This presentation will give a quick overview of the AJPF adaptations, look at some preliminary results and discuss possible uses of the system.

### 3.10 Verifying Autonomous Systems

*Michael Fisher (University of Liverpool, GB)*

License  Creative Commons BY 3.0 Unported license  
© Michael Fisher  
Joint work of Fisher, Michael; Dennis, Louise

As the internal architectures of autonomous systems become more complex, the need for their activities to be both understandable and explainable is increasing. This, combined with the development of agent model-checking techniques, is beginning to allow practical autonomous systems to be verified. Beyond straightforward analysis of functional requirements of autonomous systems, it is increasingly important to (logically) specify and verify both legal and ethical aspects. In this talk, we describe the problems (and partial solutions) associated with these aspects.

### 3.11 Resolution for Temporal Logics of Knowledge


*Michael Fisher (University of Liverpool, GB)*

License  Creative Commons BY 3.0 Unported license  
© Michael Fisher  
Joint work of Fisher, Michael; Dixon, Clare; Nalon, Claudia

I will briefly outline our work on automated proof methods, particularly clausal resolution methods, for deciding temporal logics of knowledge, belief, etc. The short talk will also touch on current implementation technology for such systems.

### 3.12 Information values in multi-agent bargaining scenarios

*Tim French (University of Western Australia – Nedlands, AU)*

License  Creative Commons BY 3.0 Unported license  
© Tim French

This talk will discuss ongoing work investigating the role information and uncertainty has in multi-agent bargaining scenarios. Game theoretic analyses of bargaining and auctions are well-established. We are interested in the interaction between the bargaining process and epistemic state of the agents involved. In one direction we may consider the question of releasing sufficient information to enable agents to find a stable bargaining solution. In the other direction we can consider the problem of determining and quantifying the amount of information an agent acquires by participating in a bargaining process. We will describe some related work and some broad goals in terms of assigning value to information, and evaluating economic processes with respect to the relative uncertainty of the participating agents.

### 3.13 The synthesis and actuation of informative events

*Tim French (University of Western Australia – Nedlands, AU)*

License  Creative Commons BY 3.0 Unported license  
© Tim French

This talk will describe some recent and ongoing work in the area of synthesising information updates, and executing informative updates through message passing systems. Particularly, James Hales has recently completed work on the synthesis of uniform action models to realise a given epistemic property within the multi-modal K. We will discuss this result, extensions and applications that it may have.

### 3.14 Combining quantitative and qualitative strategic reasoning. Part I: framework

*Valentin Goranko (Technical University of Denmark, DK)*

License  Creative Commons BY 3.0 Unported license  
© Valentin Goranko

There are several traditions in studying strategic abilities of agents to achieve objectives in multi-player games, coming from game theory, logic and computer science. Game theory studies rational behavior of players, relevant for their achievement of quantitative objectives: optimizing payoffs (e.g., maximizing rewards or minimizing cost) or, more generally, preferences on outcomes. On the other hand, logic mainly deals with abilities of players for achieving qualitative objectives of players: reaching or maintaining game states with desired properties, e.g., winning states or safe states. Put as a slogan, the former tradition is concerned with how a player can become maximally rich, or pay the least possible price, in the game, while the latter tradition – with how a player can achieve a state of ‘happiness’, or avoid reaching a state of ‘unhappiness’, in the game. Studies in computer science have involved both quantitative and qualitative objectives, usually considered separately, but

there is an increasingly active recent trend to consider games with combined objectives. We propose a logical framework, first presented in [1], combining the two traditions by enriching concurrent game models with payoffs for the normal form games associated with the states and with guards on the actions available to players in terms of their payoffs. Respectively, we propose a quantitative extension of the logic ATL\* enabling the combination of quantitative and qualitative reasoning. In the Part I of this presentation I introduce and discuss the framework. Part II of the talk, presented by Nils Bulling, discusses the model-checking problem for the Quantitative ATL\* on concurrent game models with payoffs, mentions some decidability and undecidability results and some related work.

### References

- 1 Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *Proceedings of the 1st International Workshop on Strategic Reasoning*, Electronic Proceedings in Theoretical Computer Science, pages 33–41, Rome, Italy, March 2013.

### 3.15 A few remarks about related work in Pretoria

*Stefan Gruner (University of Pretoria, ZA)*

License  Creative Commons BY 3.0 Unported license  
© Stefan Gruner

I am currently not working in the field of multi-agent systems (MAS). In a very short statement on the last day of the seminar I mentioned that I had only two older publications in the context of MAS, namely [1] [2]; however the ‘agents’ in those publications need not be ‘intelligent’ in the sense of AI. With my main interest in software engineering methodology I participated in the seminar’s discussion sub-group on the topic of a to-be-developed MAS-specific theory of software testing that must reach beyond the theory of testing for classical (non-MAS) software systems.

### References

- 1 Bilel Derbel, Mohamed Mosbah, Stefan Gruner. *Mobile Agents implementing Local Computations in Graphs*. Lecture Notes in Computer Science 5214, pp. 99-114, Springer-Verlag, 2008.
- 2 Stefan Gruner. *Mobile Agent Systems and Cellular Automata*. Journal for Autonomous Agents and Multi-Agent Systems 20/2, pp. 198-233, Springer-Verlag, 2010.

### 3.16 Yet Another Modal Notation for Strategy Contexts

*Dimitar Guelev (Bulgarian Academy of Sciences – Sofia, BG)*

License  Creative Commons BY 3.0 Unported license  
© Dimitar Guelev

I highlight an extension of ATL with knowledge [3] and both knowledge and contexts [2]. It admits some interesting instances of a proof rule which was introduced for PDL<sup>∩</sup> [1]. To make the transition to strategy-context enabled semantics, just one truly specific axiom, which is taken from [4], appears necessary. Comparison with some other existing ATL-based

notations for strategy contexts shows them to be of the same expressive power in the case of complete information.

Dimitar P. Guelev was partially supported by Bulgarian NSF Grant DID02/32/2009. He is grateful to the organizers, and especially to Bernd-Holger Schlingloff for his careful guidance.

### References

- 1 Philippe Balbiani and Dimitar Vakarelov. Iteration-free PDL with Intersection: a Complete Axiomatization. *Fundam. Inform.*, 45(3):173–194, 2001.
- 2 Dimitar P. Guelev and Catalin Dima. Epistemic ATL with Perfect Recall, Past and Strategy Contexts. In Michael Fisher, Leon van der Torre, Mehdi Dastani, and Guido Governatori, editors, *CLIMA*, volume 7486 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 2012.
- 3 Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An Alternating-time Temporal Logic with Knowledge, Perfect Recall and Past: Axiomatisation and Model-Checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.
- 4 Dirk Walther, Wiebe van der Hoek, and Michael Wooldridge. Alternating-time Temporal Logic with Explicit Strategies. In Dov Samet, editor, *TARK*, pages 269–278, 2007.

### 3.17 The grand game of testing

*Yuri Gurevich (Microsoft Research – Redmond, US)*

License © Creative Commons BY 3.0 Unported license  
© Yuri Gurevich

We present testing as a game between the Programmer and the Tester. While the game is usually more cooperative than antagonistic, Tester’s goal is different from that of Programmer. We discuss when the game is over and address Myers’s paradox: “The number of uncovered bugs in a program section is proportional to the number of discovered bugs in the section.”

### 3.18 Managing Policies and Trust

*Yuri Gurevich (Microsoft Research – Redmond, US)*

License © Creative Commons BY 3.0 Unported license  
© Yuri Gurevich

**Joint work of** Blass, Andreas; Guido de Caso; Gurevich, Yuri

**Main reference** A. Blass, G. de Caso, Y. Gurevich, “An introduction to DKAL,” Microsoft Research Tech Report MSR-TR-2012-108.

**URL** <http://research.microsoft.com/en-us/um/people/gurevich/Opera/216.pdf>

With the advent of cloud computing, the necessity arises to manage policies and trust automatically and efficiently. In a brick-and-mortar (B&M) setting, clerks learn unwritten policies from trustworthy peers. And if they don’t know a policy, they know whom to ask. In the B&M-to-cloud transition, the clerks disappear. Policies have to be explicit and managed automatically. The more challenging problem yet is how to handle the interaction of the policies of distrustful principals, especially in federated scenarios where there is no central authority. The DKAL project (Distributed Knowledge Authorization Language) was created to deal with such problems. The new language, new logics and tools keep evolving. We discuss the current state of the project.

### 3.19 Logics for Multi-Agent Systems


*Andreas Herzig (Université Paul Sabatier – Toulouse, FR)*

**License**  Creative Commons BY 3.0 Unported license  
© Andreas Herzig

I classify various logics for MAS according to the epistemic and the action dimension. I highlight problematic aspects of each of the standard accounts, including the frame problem, strategy contexts and uniform strategies.

### 3.20 Concepts, Agents, Strategies... and Coalitions. ATL goes (monadic) first order

*Wojtek Jamroga (University of Luxembourg, LU)*

**License**  Creative Commons BY 3.0 Unported license  
© Wojtek Jamroga

**Main reference** W. Jamroga, “Concepts, Agents, and Coalitions in Alternating Time,” in Proc. of the 20th European Conf. on Artificial Intelligence (ECAI’12), Frontiers in Artificial Intelligence and Applications, Vol. 242, pp. 438–443, IOS Press, 2012.

**URL** <http://dx.doi.org/10.3233/978-1-61499-098-7-438>

**URL** <http://icr.uni.lu/wjamroga/papers/at1+dl12ecai.pdf>

I consider a combination of the strategic logic ATL with the description logic ALCO. In order to combine the logics in a flexible way, I assume that every individual can be (potentially) an agent. I also take the novel approach to teams by assuming that a coalition has an identity on its own, and hence its membership can vary. In terms of technical results, I show that the logic does not have the finite model property, though both ATL and ALCO do. I conjecture that the satisfiability problem may be undecidable. On the other hand, model checking of the combined logic is decidable and even tractable. Finally, I define a particular variant of realizability that combines satisfiability of ALCO with model checking of the ATL dimension, and I show that this new problem is decidable.

#### References

- 1 Wojciech Jamroga. *Concepts, Agents, and Coalitions in Alternating Time*. Proceedings of the 20th European Conference on Artificial Intelligence ECAI 2012, pp. 438–443, IOS Press, 2012.

### 3.21 ATL with strategy contexts – part 1

*François Laroussinie (Université Paris-Diderot – Paris, FR)*

**Joint work of** Laroussinie, François; Markey, Nicolas  
**License**  Creative Commons BY 3.0 Unported license  
© François Laroussinie

ATLsc is an extension of ATL with strategy contexts: the agents are committed to their strategies during the evaluation of formulas. This makes a huge difference with standard ATL. In this first talk, we will discuss the expressive power of ATLsc. In particular we will give several examples of properties that can be expressed with this formalism.

### 3.22 ATL with strategy contexts – part 2

Nicolas Markey (*ENS Cachan, FR*)

**Joint work of** Laroussinie, François; Markey, Nicolas  
**License** © Creative Commons BY 3.0 Unported license  
 © Nicolas Markey

This second talk (the first one is proposed by François Laroussinie) will focus on the decision procedures for ATLsc and its complexity (for model checking and satisfiability). For this we will use an extension of CTL with quantifications over atomic propositions (QCTL). Indeed we will see that (1) verification problems for ATLsc (or Strategy Logic) can be naturally reduced to a problem over QCTL, and (2) decision procedures can be described in a simpler way for QCTL.

### 3.23 Uniform Strategies

Bastien Maubert (*Université de Rennes 1, FR*)

**License** © Creative Commons BY 3.0 Unported license  
 © Bastien Maubert  
**Joint work of** Maubert, Bastien; Pinchinat, Sophie  
**Main reference** B. Maubert, S. Pinchinat, L. Bozzelli, “The Complexity of Synthesizing Uniform Strategies,” in Proc. of the 1st Int’l Workshop on Strategic Reasoning, EPTCS, Vol. 112, pp. 115-122, 2013.  
**URL** <http://dx.doi.org/10.4204/EPTCS.112.17>

We study a general notion of uniform strategies that subsumes several existing notions of strategies subject to some uniformity constraints, like for example in games with imperfect information or model checking games for dependence logic. We present a logical language to specify such uniformity constraints. This language is basically LTL augmented with a knowledge-like operator  $R$ , where  $R\varphi$  means that  $\varphi$  holds in all related plays. One particularity of this work concerns the semantics of the  $R$  modality. Instead of choosing a specific relation over plays, like synchronous perfect-recall for example, we allow for any binary rational relation. This class of relations is very general, and in particular it contains all relations classically used in games with imperfect information and logics of knowledge and time (perfect/imperfect recall, synchronous/asynchronous...). Rational relations are recognized by finite state transducers, which allows us to study the decidability and complexity of synthesizing uniform strategies for different subclasses of rational relations. Our results imply the decidability of the model checking of LTLKn with asynchronous perfect recall, and more generally we have that the strategy problem in games with a winning condition expressed in LTLK is decidable as long as the relation that represents the knowledge is rational.

### 3.24 A Poor Man’s Technique for Reasoning About Knowledge

Stephan Merz (*LORIA – Nancy, FR*)

**License** © Creative Commons BY 3.0 Unported license  
 © Stephan Merz

Representing and reasoning about knowledge is fundamental for modeling and analyzing multi-agent systems. Several logics have been proposed that combine epistemic and temporal

or dynamic operators, and that support reasoning about the state of knowledge of agents based on the information they observe, e.g. resulting from updates due to point-to-point communication or broadcasts. Specialized model checkers such as DEMO [3] implement efficient procedures for evaluating formulas written in logics such as Dynamic Epistemic Logic (DEL, [1]). We report on an experiment on encoding the semantics of DEL directly in a constraint solver and using that encoding for solving the well-known “Sum and Product” puzzle [2]. The models are written in TLA<sup>+</sup> and are evaluated using the constraint solving techniques implemented in ProB [4]. The running times compare very favorably with those reported for DEMO, indicating that general-purpose solvers may in certain cases be quite competitive, at least for prototyping verification engines for new logics.

### References

- 1 H. van Ditmarsch, W. van der Hoek, B. Kooi. Dynamic Epistemic Logic. Synthese Library 337. Springer (2007).
- 2 H. van Ditmarsch, J. Ruan, R. Verbrugge. Sum and Product in Dynamic Epistemic Logic. *J. Log. Comput.* 18(4): 563-588 (2008).
- 3 Jan van Eijck. DEMO – A Demo of Epistemic Modelling. In: *Interactive Logic. Proc. 7th Augustus de Morgan Workshop* (J. van Benthem, D. Gabbay, B. Löwe, eds.) *Texts in Logic and Games* 1:305–363 (2007).
- 4 M. Leuschel, M. J. Butler: ProB: an automated analysis toolset for the B method. *STTT* 10(2): 185-203 (2008)

## 3.25 Reasoning About Strategy

*Aniello Murano (University of Napoli, IT)*

**License** © Creative Commons BY 3.0 Unported license

© Aniello Murano

**Joint work of** Mogavero, Fabio; Perelli, Giuseppe ; Sauro, Luigi; Vardi, Moshe

**Main reference** F. Mogavero, A. Murano, M.Y. Vardi, “Reasoning About Strategies,” in *Proc. of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS’10)*, LIPIcs, Vol. 8, pp. 133–144, Schloss Dagstuhl, 2010.

**URL** <http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2010.133>

In open systems verification, to formally check for reliability, one needs an appropriate formalism to model the interaction between agents and express the correctness of the system no matter how the environment behaves. An important contribution in this context is given by modal logics for strategic ability, in the setting of multi-agent games, such as ATL, ATL\*, and the like. In this talk, I will introduce Strategy Logic as a powerful logic framework for reasoning explicitly about strategies in multi-agent concurrent games. As a key aspect, SL uses first-order quantifications over strategies, where strategies are not glued to a specific agent, but an explicit binding operator allows to bind an agent to a strategy variable. This allows agents to share strategies or reuse one previously adopted. In this talk, I will discuss about the model checking and the satisfiability decision problems for SL and show that they are undecidable and NonElementarySpace-hard, respectively. This negative result has successfully spurred us to investigate syntactic fragments of SL, strictly subsuming ATL\*, with elementary decision problems. In this talk I will present some of these fragments and discuss their properties.



### 3.26 Bounded model checking for LTLK

Wojciech Penczek (*Siedlce University of Natural Sciences and Humanities, PL*)

License  Creative Commons BY 3.0 Unported license  
© Wojciech Penczek

We present a novel approach to verification of multi-agent systems by bounded model checking for Linear Time Temporal Logic extended with the epistemic component (LTLK). The systems are modelled by two variants of interpreted systems: standard and interleaved ones. Our method is based on binary decision diagrams (BDD). We describe the algorithm and provide its experimental evaluation together with the comparison with another tool. This allows to draw some conclusions concerning which semantics is preferable for bounded model checking LTLK properties of multi-agent systems.

### 3.27 Abstract planning using genetic algorithms

Wojciech Penczek (*Siedlce Univ of Natural Sciences and Humanities, PL*)

License  Creative Commons BY 3.0 Unported license  
© Wojciech Penczek

**Joint work of** Niewiadomski, Artur; Penczek, Wojciech; Skaruz Jaroslaw

**Main reference** A. Niewiadomski, W. Penczek, J. Skaruz, “Towards automated abstract planning based on a genetic algorithm,” Technical Report 1026, ICS PAS, Warsaw, 2012.

**URL** <http://artur.ii.uph.edu.pl/papers/rep1026.pdf>


The lecture is based on joint work [1], which discusses a new approach based on nature inspired algorithms to an automated abstract planning problem, which is a part of the web service composition problem. An abstract plan is defined as an equivalence class of sequences of service types that satisfy a user query. Intuitively, two sequences are equivalent if they are composed of the same service types, but not necessarily occurring in the same order. The objective of our genetic algorithm (GA) is to return representatives of abstract plans without generating all the equivalent sequences. The lecture presents experimental results, which show that GA finds solutions for very large sets of service types in a reasonable time.

#### References

- 1 A. Niewiadomski, W. Penczek, and J. Skaruz. Towards automated abstract planning based on a genetic algorithm. Technical Report 1026, ICS PAS, 2012. <http://artur.ii.uph.edu.pl/papers/rep1026.pdf>.

### 3.28 Tools for MAS verification: where do we go next?

Franco Raimondi (*Middlesex University – London, UK*)

License  Creative Commons BY 3.0 Unported license  
© Franco Raimondi

A number of software tools are available for MAS verification, and their performance has improved by orders of magnitude in the past decade. However, most tools have their own input language and often specialize in one verification technology, or only support checking a specific type of property. As a result, the adoption of MAS verification tools is still very limited in the “real world”. In this presentation we suggest a different approach to MAS

verification: tools should be moved closer to real systems by means of (re-usable) connectors and libraries, and should be seen as components of a more general framework. We provide an example of this framework using the Brahms modelling language for MAS, and various model checkers to perform verification.

### 3.29 Doomsday Equilibria for Omega-Regular Games


*Jean-François Raskin (Université Libre de Bruxelles, BE)*

**License**  Creative Commons BY 3.0 Unported license  
© Jean-François Raskin

Two-player games on graphs provide the theoretical framework for many important problems such as reactive synthesis. While the traditional study of two-player zero-sum games has been extended to multi-player games with several notions of equilibria, they are decidable only for perfect-information games, whereas several applications require imperfect-information games. In this paper we propose a new notion of equilibria, called doomsday equilibria, which is a strategy profile such that all players satisfy their own objective, and if any coalition of players deviates and violates even one of the players objective, then the objective of every player is violated. We present algorithms and complexity results for deciding the existence of doomsday equilibria for various classes of  $\omega$ -regular objectives, both for imperfect-information games, as well as for perfect-information games. We provide optimal complexity bounds for imperfect-information games, and in most cases for perfect-information games.

### 3.30 Specification based testing in an institutional setting

*Markus Roggenbach (Swansea University, UK)*

**License**  Creative Commons BY 3.0 Unported license  
© Markus Roggenbach

**Joint work of** Phillip James; Faron F Moller; Hoang Nga Nguyen; Markus Roggenbach; Steve Schneider; Helen Treharne

**Main reference** F. Moller, H.N. Nguyen, M. Roggenbach, S. Schneider, H. Treharne, “Defining and Model Checking Abstractions of Complex Railway Models Using CSP||B,” in Proc. of the Haifa Verification Conference (HVC’12), LNCS, Vol. 7857, pp. 193–208, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-39611-3\\_20](http://dx.doi.org/10.1007/978-3-642-39611-3_20)

**Main reference** P. James, M. Trumble, H. Treharne, M. Roggenbach, S. Schneider, “OnTrack: An Open Tooling Environment for Railway Verification,” in Proc. of the 5th Int’l NASA Symp. on Formal Methods, LNCS, Vol. 7871, pp. 435–440, Springer, 2013.

**URL** [http://dx.doi.org/10.1007/978-3-642-38088-4\\_30](http://dx.doi.org/10.1007/978-3-642-38088-4_30)

It is becoming common industrial practice to utilize Domain Specific Languages (DSLs) for designing systems. Such DSLs offer constructs native to the specific application area. Formal methods often fail to be easily accessible for engineers, but designs formulated in DSLs are open for systematic and, possibly, automated translation into formal models for verification.

In this talk, we show that DSLs also allow abstractions to be formulated at the domain level. We demonstrate on the example of the Railway domain that (1) such abstractions exists over the boundary of different specification languages (CSP, CSP||B, CASL) and (2) and demonstrate by the means of our tool OnTrack how to support & automatize such abstractions.

### 3.31 Efficient Testing of Software Product Lines

*Ina Schaefer (TU Braunschweig, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Ina Schaefer

**Joint work of** Schaefer, Ina; Malte Lochau; Sascha Lity

**Main reference** M. Lochau, I. Schaefer, J. Kamischke, S. Lity, “Incremental Model-Based Testing of Delta-Oriented Software Product Lines,” in Proc. of the 6th Int’l Conf. on Tests and Proofs (TAP’12), LNCS, Vol. 7305, pp. 67–82, Springer, 2012.

**URL** [http://dx.doi.org/10.1007/978-3-642-30473-6\\_7](http://dx.doi.org/10.1007/978-3-642-30473-6_7)

Testing software product lines by considering each product variant in isolation is impracticable due to the high number of potential product configurations. Therefore, applying reuse principles also to test artifacts in a concise way for efficient testing is essential. In this talk, I address this open issue by presenting a novel, model-based SPL testing framework based on reusable test models and incremental test suite evolution. Test artifacts are incrementally evolved for every product variant by explicitly considering commonality and variability between two subsequent products under test. I illustrate the framework by means of an automotive case study and compare our experimental results with alternative SPL testing strategies with respect to efficiency improvements.

### 3.32 On the specification and analysis of contracts (normative texts)

*Gerardo Schneider (University of Gothenburg, SE)*

**License** © Creative Commons BY 3.0 Unported license  
© Gerardo Schneider

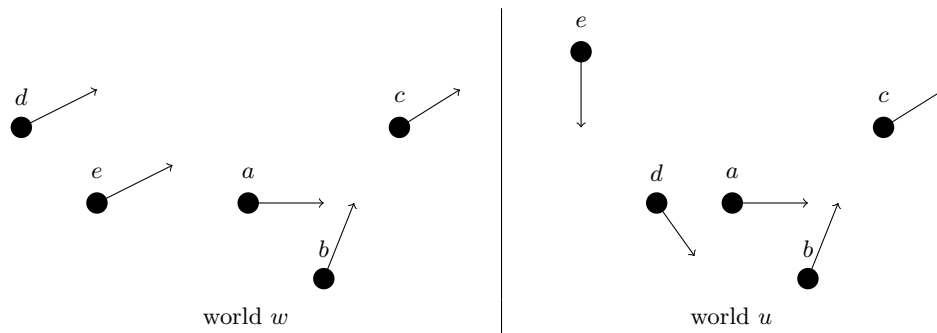
Our general aim is to be able to provide a formal language to specify and analyze normative texts in general, and electronic contracts in particular. In this talk we introduce the idea of a framework where normative texts could be translated into a Controlled Natural Language (CNL) and then into a formal language, in order to be analyzed both statically and at runtime. As a step towards such an ambitious aim, we present AnaCon, a framework where a restricted version of normative texts are written in a CNL and automatically translated into the formal language CL using the Grammatical Framework (GF). In AnaCon such CL expressions are analyzed for normative conflicts (i.e., whether there are conflicting obligations, permissions and prohibitions) by the tool CLAN which gives a counter-example in case a conflict is found. We finally discuss research challenges and future directions in the area.

### 3.33 Flatland logic

*François Schwarzentruber (ENS Cachan Brittany extension – Rennes, FR)*

**License** © Creative Commons BY 3.0 Unported license  
© François Schwarzentruber

There are possible needs in applications such as video games for reasoning about knowledge and perception of agents. Unfortunately the behaviour of artificial agents is still nowadays often described using imperative languages such as JAVA (or script languages such as Lua). But the use of knowledge programs is a high-level option. We propose a grounded variant of Dynamic epistemic logic called Flatland logic where we can express properties about what



■ **Figure 1** Two possible 2-dimensional worlds that are indistinguishable for agent  $a$ .

agents perceive and know about the world. The semantics is built on top of a Kripke model. A possible world in the Kripke model is a mapping that assigns a location to each agent in the space. An agent sees a half-space. Two worlds are indistinguishable for agent  $a$  if, and only if, agent  $a$  sees the same thing in both worlds. Figure 1 shows two possible worlds  $w$  and  $u$  that are indistinguishable for agent  $a$ . For instance, agent  $d$  sees  $e$  in world  $w$  and agent  $d$  does not see  $e$  in world  $u$ . Agent  $a$  knows that agent  $b$  sees agent  $c$ .

We then give results about its axiomatisation and the complexity of its model checking and satisfiability problems. In the one-dimensional case, agents are placed on a line. The one-dimensional case is axiomatized and the corresponding model checking and satisfiability problems are both PSPACE-complete. Concerning the two-dimensional case, we do not know whether there exists a finite axiomatization. We know that both the corresponding model checking and satisfiability problems are decidable but we do not know the exact complexity. There are many open issues concerning implementation and expressiveness of the logic.

### 3.34 Before the announcement

*Hans van Ditmarsch (LORIA – Nancy, FR)*

License Creative Commons BY 3.0 Unported license  
© Hans van Ditmarsch

This concerns ongoing (but so far very tentative) work with Andreas Herzig and Philippe Balbiani. The well-known logic of public announcement models the effect of public events on information states:  $[\varphi]\psi$  stands for ‘after public announcement of  $\varphi$ ,  $\psi$  is true’. Suppose you want to go in the other direction:  $[\varphi]^c\psi$  stands for ‘before the announcement of  $\varphi$ ,  $\psi$  was true’. What is the logic of that? For example,  $[p]^cp$  is valid: before (truthful) public announcement of  $p$ ,  $p$  must already have been considered possible. This logic is quite different from the history operators in the work of Joshua Sack, where one goes back one step in time, given history-based structures. Instead, in this ‘before the announcement’ logic, any structure with ‘more uncertainty’ than the current state can have led, after an announcement, to that current state of information. So in that sense the  $[\varphi]^c$  operator has aspects of propositional quantification. Dual to the ‘refinement quantifier’ in ‘Refinement Modal Logic’ by Bozzelli et al., we seem to be looking in this case for a ‘simulation quantifier’:  $[\varphi]^c\psi$  is true in state  $s$  of model  $M$  now, if  $\psi$  is satisfied in any *simulation* of  $(M, s)$ ... plus something else. Simulation quantifiers would be more proper for ‘before the event’ logic, the dual of event model (action model) logic. The case of ‘before the announcement’ is more restricted.

### 3.35 Scaling up Test Data Generation

*Ramanathan Venkatesh (Tata Consultancy Services – Pune, IN)*

**License** © Creative Commons BY 3.0 Unported license  
© Ramanathan Venkatesh

**Joint work of** R Venkatesh; Anand Yeolekar; Divyesh, Unadkat

**Main reference** A. Yeolekar, D. Unadkat, V. Agarwal, S. Kumar, R. Venkatesh, “Scaling Model Checking for Test Generation using Dynamic Inference,” in Proc. of the 2013 IEEE 6th Int’l Conf. on Software Testing, Verification and Validation (ICST’13), pp. 184–191, IEEE, 2013.

**URL** <http://dx.doi.org/10.1109/ICST.2013.29>

Structural test coverage criteria are very effective in finding bugs and also required by standards such as DO 178B. Model checkers can be used to generate test data to achieve the required coverage but model checkers unfortunately do not scale up to industry size code. To address this problem of scalability we combine dynamic analysis with model checking. We employ dynamic analysis to determine a pre-/post- condition pair for complex functions. Then we use a model checker after replacing complex functions by their pre-/post- conditions. This technique has given us much better scalability than using plain model checkers.

### 3.36 Show me your friends and I tell you who you are

*Karsten Wolf (Universität Rostock, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Karsten Wolf

For an open system  $S$  (e.g. a MAS), and a desired property  $\varphi$ , a  $\varphi$ -partner is another open system such that the composition  $S + P$  satisfies  $\varphi$ . For many properties  $\varphi$ , we can compute a finite characterization of the (typically infinite) set of  $\varphi$ -partners. It can be used for several interesting applications:

- Safe exchange of  $S$  by  $S'$  is possible if  $\text{partners}(S)$  is a subset of  $\text{partners}(S')$ ; we can decide that using the finite characterization
- From the set of partners, test cases may be selected; the characterization offers some notion of partner coverage
- Correction of  $P$  in a collaboration with  $S$  is possible: Select, among the partners of  $S$ , the one that is most similar to  $P$ ; the finite characterization helps to reason about the infinite set of candidates.

In the talk, we sketch the work we have done and address some challenges in the MAS area, e.g. declarative representations of internal state.

### 3.37 Synthesis of Knowledge-Based Program Implementations

*Ron van der Meyden (University of New South Wales – Sydney, AU)*

**License** © Creative Commons BY 3.0 Unported license  
© Ron van der Meyden

Knowledge-based programs are a representation of agent behaviour, in which agent’s actions are conditioned on formulas expressing properties of the agent’s knowledge. This provides a useful level of abstraction that yields protocol descriptions that are independent of assumptions

about the environment in which the protocol runs, and disentangles the question of what the agent needs to know in order to perform its task from the questions of how it obtains and represents that knowledge. On the other hand, knowledge-based programs are more like specifications than like executable code. To execute a knowledge-based program in a concrete environment it is necessary to determine conditions on the agent’s local state that are equivalent to the knowledge properties.

The talk reviewed early work on the topic, which studied the computational complexity of finding implementations, and then addressed the question of how implementation may be found in practice. Two approaches were discussed: a partially automated approach based on epistemic model checking [3, 1, 2] and current work that aims to develop a fully automated approach based on symbolic representations of knowledge [4].

## References

- 1 Omar I. Al-Bataineh and Ron van der Meyden. Epistemic model checking for knowledge-based program implementation: An application to anonymous broadcast. In Sushil Jajodia and Jianying Zhou, editors, *SecureComm*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 429–447. Springer, 2010.
- 2 Omar I. Al-Bataineh and Ron van der Meyden. Abstraction for epistemic model checking of dining cryptographers-based protocols. In Krzysztof R. Apt, editor, *TARK*, pages 247–256. ACM, 2011.
- 3 Kai Baukus and Ron van der Meyden. A knowledge based analysis of cache coherence. In Jim Davies, Wolfram Schulte, and Michael Barnett, editors, *ICFEM*, volume 3308 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2004.
- 4 Xiaowei Huang and Ron van der Meyden. Symbolic synthesis of knowledge-based program implementations with synchronous semantics. pages 121–130, 2013.

## 4 Working Groups

### 4.1 Case Study Description: Elevators

*Working Group 2*

License © Creative Commons BY 3.0 Unported license  
© Working Group 2

Real-Life setting: Elevators in skyscrapers with decentralized control. We assume that there is a group of  $n$  elevators serving  $m$  floors. All lift controllers (and, hence, the strategies of all lifts) are essentially the same, except for the following configuration parameter: Not all elevators stop at all floors (the elevators can travel at different speeds). Passengers arrive at the different floors at an unpredictable rate.

#### 4.1.1 What can passengers do?

**Variante 1:** Passengers have a “call” button to call for an elevator.

**Variante 2:** Passengers have “up” and “down” buttons indicating the direction where they want to go.

**Variante 3:** Passengers have a “request” button for each floor indicating the floor they want to reach.

In Variant 1) and 2), each elevator has an internal set of buttons where passengers can indicate where they want to go.

**Variant 4:** On each floor, there is a sensor sensing how many people are waiting at that floor.

**Variant 5:** There may be a time-dependent scheme according to which elevators are being requested, e.g., before 9 am it may be mainly upwards and after 5 pm it may be mainly downwards.

**Variant 6:** Certain requests may have to be served with high priority, e.g., the president arriving at the park deck always has to find a lift waiting for him. Other high-priority calls may be for firefighter mode and emergency journeys.

#### 4.1.2 What can elevators do?

Each lift knows its current position/direction and notices all passenger requests. The elevators have to communicate and negotiate which requests to serve in which order. Each elevator has a decentralized control of its own and can communicate with the others:

**Variant 1:** via broadcast,

**Variant 2:** to each other lift individually,

**Variant 3:** to “adjacent” lifts (where the adjacency relation is to be defined).

Each elevator can communicate the following information:

- its own position,
- its own direction,
- maybe also its current plan about floors to serve.

The lift may be able to predict the calls or may move to a “standard” position when idle.

#### 4.1.3 Goals:

The overall goal is to serve all passengers “fairly” (task: define what that means), with a minimal number of moves (or equivalently, a minimal amount of energy) and/or a minimal waiting time. An additional requirement is that there should not be a single request not served for a certain amount of time, e.g., 10 min.


Goals for the individual elevator are:

- It could be lazy, e.g., want to move as little as possible;
- or it could be eager, trying to serve as many requests as possible;
- or optimize the travel according to some other criterion.
- Variant: There may exist malicious elevators, e.g., an elevator may be faulty or send false messages.

Each elevator should be able to reason about his decisions, e.g., there are cost and reward functions to support the argument. A cost function could be, e.g., the number and distance of moves, fast travel is more expensive than slow travel, etc. A reward could also be given for each passenger served.

## 4.2 Case Study Description: Access Control

*Dimitar P. Guelev*

License  Creative Commons BY 3.0 Unported license  
© Dimitar P. Guelev

### 4.2.1 Description:

An automated conference review system, for being well understood infrastructure in academia.

**Agents:** The system designer, the program chairs, the programme committee members, the authors of submissions, some external reviewers. These groups need not be all disjoint.

**Actions:** Those which take place through the automated system, subject to history-dependent permissions, include appointing PC members, submitting papers, assigning papers to review, uploading reviews, various queries, etc.

**Goals:** The *system designer* (see also Section 4.2.3 *Challenge* below) must select permission conditions which implement common express requirements on the reviewing process such as *anonymity of reviewing* and avoid recognized forms of *conflict of interest*, e.g., an author cannot interfere in the reviewing of her own submissions, and, unless a PC member, can follow only her own submissions; a PC decision normally requires a full set of reviews, etc. Overly restricted access may cause unwanted *unease*.

For (groups of) agents who are modelled *in the system*, goals include:

1. The PC want to collect the due sets of reviews and reach decisions within the promised deadlines.
2. Everyone is interested in influencing the conference programme.
3. Everyone is interested in broadening their grasp on the reviewing process as much and as soon as possible.
4. (Un)deniability.

### 4.2.2 Comments:

The scenario is partly *collaborative* and partly *competitive*. Goal 1 is collaborative, but may face shortage of external subreviewers and thus fail *independent reviewing*. Goals of group 3 are typically competitive and relate to inferred knowledge, which can jeopardize, e.g. anonymity. Goals 2 are typically in the reach of PC members, with the appointment of subreviewers in their powers. Goal 4 is primarily competitive; its collaborative aspects are by far less trivial.

### 4.2.3 Challenge:

Analyses must assist the designer in choosing, maintaining and evolving implementations of the system such that:

- each agent, possibly jointly with other agents, can achieve his or her legitimate goals by following preferably simple guidelines, and
- no schemes for reaching illegitimate goals (as understood for peer-reviewed conferences) are conceivable.

The scenario has been used previously, to illustrate research on and test algorithms for verifying access control systems in [1, 3, 2] and elsewhere, in a form that enables clear cut correspondence with the technical matter therein.



## References

- 1 Dimitar P. Guelev, Mark Ryan, and Pierre-Yves Schobbens. Model-Checking Access Control Policies. In Kan Zhang and Yuliang Zheng, editors, *ISC*, volume 3225 of *Lecture Notes in Computer Science*, pages 219–230. Springer, 2004.
- 2 Masoud Koleini, Eike Ritter, and Mark Ryan. Model Checking Agent Knowledge in Dynamic Access Control Policies. In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2013.
- 3 Nan Zhang, Mark Ryan, and Dimitar P. Guelev. Synthesising verified access control systems through model checking. *Journal of Computer Security*, 16(1):1–61, 2008.

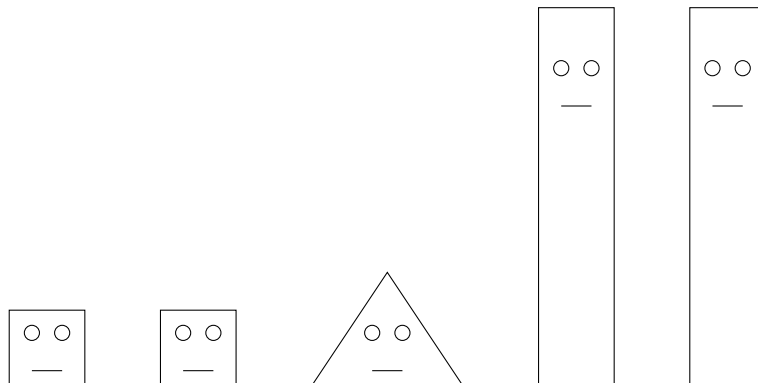
## 4.3 Case Study Description: Earthquake Rescue Mission

### Working Group 1

License © Creative Commons BY 3.0 Unported license  
© Working Group 1

#### 4.3.1 Description of the example

An earthquake occurs in a city, making many casualties, and incidentally destroying all the Internet connections. Robots are sent for a rescue mission.



Instead of sending a single type of robots that would be able to achieve all the necessary tasks (it would be too big/complicated, and some tasks may require opposite features), it seems reasonable to use robots of various kinds, with different capabilities/roles (communications, observations/detections/perception, path clearing, lifting, etc.), and also to send multiple robots of each kind, in order to parallelize the rescue mission.

Relevance for using MAS techniques:

- Dangerous environment: not safe to send humans do the job
- Poor communications: no centralized approach
- Diversity of tasks to perform
- Time pressure

#### 4.3.2 Modeling as a MAS

Each robot is an autonomous agent. We describe the features of such an agent.

##### Actions

- observe

- send/receive messages
- move
- lift
- clean
- collect

**Imperfect information** The robots may have a map in memory, but it may be inaccurate (the earthquake modifies the area) or local (limited to what they observe/learn from other robots). Sensors may be unreliable (robots may be damaged). This leads to:

- Uncertainty concerning its own location
- Uncertainty concerning the health of human casualties

#### Goals

- One main goal: rescue people.
- Subgoals, possibly dynamically chosen: look for human beings, route finding and cleaning, establish communications, agreements among a coalition of robots, inform other robots (position of a victim...), find human assistants...

**Human interaction** Robots should be able to interact with humans they meet during the rescue (basic health diagnosis, take orders from survivors...).

**Testing and verification** Difficult to define the goal (to rescue the maximum number of victims?)

## 4.4 Case Study Description: Examples from dialog/social systems

### *Working Groups 3 and 4*

**License** © Creative Commons BY 3.0 Unported license  
© Working Groups 3 and 4

We explore the possibility of testing the four following application domains with MAS techniques: dialog systems, social networks, stability and resilience analysis in social systems and stock markets.

Tasks:

- Find existing benchmarks
- For each topic achieve the following.
  - Develop a concrete example
  - Give a description taking the following features into account

Uncertainty:

- more than 1 agent
- strategies involved
- autonomous control
- objectives and intentions
- communication
- information (K&B)
- human interaction

#### 4.4.1 Previous Benchmarks & Competitions

What is a benchmark? A concrete scenario in which the agents compete against each other with well defined goals.

Can we borrow from Benchmarks for Security Protocols? They are so big and so real that it is hard to test the proposal. Possibility of having good enough simulations to test systems against them.

**Trading Agent Competition:** Designed to promote and encourage high quality research into the trading agent problem. Currently uses two scenarios: TAC Classic, a “travel agent” scenario based on complex procurement on multiple simultaneous auctions, and TAC SCM, a PC manufacturer scenario based on sourcing of components, manufacturing of PC’s and sales to customers.

<http://tac.sics.se/page.php?id=1>

Comments/Critics: Use of machine learning.

**General Game Playing Description:** General Game Playing is a project of the Stanford Logic Group of Stanford University, California, which aims at creating a platform for general game playing. The games are defined by sets of rules represented in the Game Description Language. In order to play the games, players interact with a game hosting server that monitors moves for legality and keeps players informed of state changes. Since 2005, there have been annual General Game Playing competitions at the AAAI Conference. The winner of the competition is awarded with US\$10,000.

[http://en.wikipedia.org/wiki/General\\_game\\_playing](http://en.wikipedia.org/wiki/General_game_playing)

**Robocup Rescue Description:** The intention of the RoboCupRescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive systems in future.

<http://www.robocup.org/robocup-rescue/>

**Generating Instructions in Virtual Environments – GIVE:** Systems should guide a human user to navigate a virtual environment to a certain location. The systems generate natural language instructions, but they can monitor the actions taken by the user. The user cannot make queries, only navigate the environment.

<http://www.give-challenge.org/research/>

Comments/Critics: No real use (yet) of knowledge or beliefs. Main concern “grounding” (connected to common knowledge, but never made explicit). But systems need to include some modelling of the user. All systems include a planning component. No SAT or Model Checking involved. Reasoning is compiled in the actions taken.

General qualities of existing agents:

- High connectivity
- Low level of reasoning
- General decisions for the definition of examples.
- Qualitative vs. Quantitative
- Specify agents
- Specify actions
- Specify goals

We now describe five proposals of concrete examples for the domains of applications considered.

#### 4.4.2 Property Market

Description: The general framework is that of buying and selling houses. Buyers have preferences (certain area) and limitations (amount of money).

**Agents:**

- Buyers,
- Sellers,
- Estate agents,
- Banks

**Goals:**

- Sellers want to get higher price
- Buyers have a private utility
- Banks want to give a loan to the most reliable buyer

**Actions:**

- Announce a price
- Bidding for a house
- Securing a loan (for a Buyer)
- Give a loan (for a Bank)
- Negate a loan (for a Bank)

**Type:**

- Qualitative Information: Give the largest possible loan to prospective worthy agents.
- Quantitative Information: Location, price

**Prediction:**

- Creation of a bubble
- Equilibria Happiness

**Comments:**

- No clear separation between buyers and sellers (a buyer might be a seller)
- State agent misinformation
- Intervention: the state intervenes to fix problems (change payoff, include taxes). They can fix the problem or make it harder.

#### 4.4.3 Concert Going

Description: A concert is being announced in a given location. People can announce whether they expect to go or not (e.g., via their facebook pages). Other agents (e.g., hotel and restaurants in the area of the concert) have access to the information and can use it for their planning. (e.g., do I go or not in the end?, how much food do I order, how much do I charge for parking?).

**Agents:**

- Concert goers
- Restaurants
- Hotels

**Goals:**

- Restaurants do not run out of food
- Concert goer: I finally go only if some other people actually go.

**Actions:**

- Announce going to the concert
- Check

**Comments:**

- Quite Complex
- Multi-Agent planning problem
- Friends provide network: the network is not accessible to everybody

Perspective of the city hall: Cooperatively make the system reach some goal.

**4.4.4 Stock Market**

Description: Only one stock

**Agents:** Finite number of agents not determined.

**Actions:**

- Sell
- Buy

**Aim:** Have a system that prevents certain problems from occurring. Intervention. Relation to Thomas Ågotnes' ideas on social laws.

**Comments:** Mainly qualitative, no probabilities. But includes quantities (price of the stock).

**4.4.5 Book Prices**

Description: book buying/selling, with the constraint (legal obligation) that the prices must remain below a certain threshold. Assumption: unlimited supply of books.

**Agents:**

- Book sellers
- Book buyers

**Goal:** Sell the books for the best possible price

**Actions:**

- Buy a book
- Raise the price of the book
- Look at the price of a book

**Beliefs:** They don't have access to the actual demand

**Comment:** A step towards the complex one.

**4.4.6 Really asymmetric example**

Description: The set up of the GIVE challenge.

**Agents:**

- Human following instructions
- System giving instructions

**Actions:**

- Navigation actions (for the User)
- Instruction generation (for the System)
- User Monitoring (for the System)

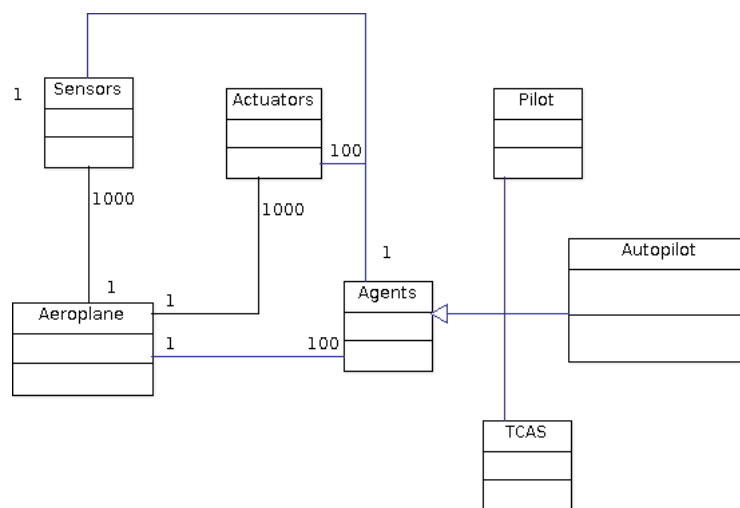
## 4.5 Case Study Description: Avionic scenario

*Franco Raimondi*

License © Creative Commons BY 3.0 Unported license  
© Franco Raimondi

### 4.5.1 Description of scenario and Modeling as MAS

An aeroplane is composed of a number of agents, such as one or more (human) pilots, autopilot, Traffic Collision Avoidance System (TCAS), etc. Also, a number of sensors are present, and each agent has access to a subset of these. Sensors may be noisy and could fail. A number of actuators are present as well, aeroplanes are coordinated by Air Traffic Controllers (ATC) on ground. ATCs have access to sensors, actuators, and other agents (such as the ground equivalent of TCAS). Figure 2 represents the aeroplane diagram, and Figure 3 represents the ATC diagram (100 means “a good amount of”, and 1000 means “a lot”).



■ **Figure 2** Diagram of an aeroplane.

### 4.5.2 State of the art

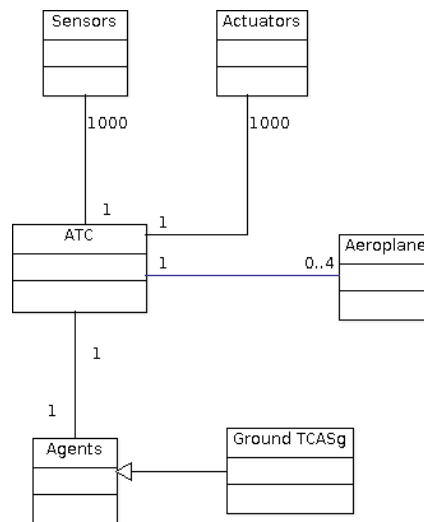
Some tools are available to model similar scenarios. See

1) J. Hunter, F. Raimondi, N. Rungta, R. Stocker, A Synergistic and Extensible Framework for Multi-Agent System Verification, to appear in Proceedings of AAMAS 2013

2) N. Rungta, G. Brat, W. Clancey, C. Linde, F. Raimondi, Chin S. and M. Shafto, Aviation Safety: Modeling and Analyzing Complex Interactions between Humans and Automated Systems, in Proceedings of ATACCS 2013

Papers available at <http://www.rmnd.net/publications/>

These papers use Brahms, see <http://www.dagstuhl.de/Materials/Files/07/07122/07122.SierhuisMaarten.Other.pdf>



■ **Figure 3** Diagram of an ATC.

### 4.5.3 Relevant requirements for verification

There are a number of possibilities. Check that safety rules are respected even in presence of noisy sensors:

- Autopilot needs to be engaged while landing in fog, and pilot has a strategy to achieve this
- What is the probability of reaching a certain state?
- If the autopilot is not engaged, the pilot knows it
- If the pilot believes that a sensor is not functioning, then the pilot has a strategy to deploy a backup sensor, etc.

Other possibilities include the encoding of interesting properties from existing procedures or from proposed standards (see NextGen and SESAR, available respectively at [http://en.wikipedia.org/wiki/Next\\_Generation\\_Air\\_Transportation\\_System](http://en.wikipedia.org/wiki/Next_Generation_Air_Transportation_System) and [http://en.wikipedia.org/wiki/Single\\_European\\_Sky\\_ATM\\_Research](http://en.wikipedia.org/wiki/Single_European_Sky_ATM_Research) for examples of Air Traffic Management systems).

### 4.5.4 Existing tools

The source code of a very simple example using MCMAS (supports CTLK+ATL) is provided below. The scenario encodes take-off and landing of an aeroplane in random weather conditions (fog, wind, clear). If the weather is foggy, the autopilot should be engaged while landing, but it should be disengaged in case of wind. It is possible to check these properties and an epistemic one such as “if it is windy, the pilot knows it” (false, because a noisy wind sensor is modelled). Also, it is possible to check ATL properties such as “the pilot has a strategy to keep the autopilot on” (false because the aeroplane could be stuck at gate). Figure 4 shows some examples of formulas model checked on the scenario example using MCMAS. When a formula is not true on a model, MCMAS can give an example of behaviour that does not verify it (see Figure 5).

```

Agent Environment
  Vars:
    aeroplane1_status : {atgate, taxiing, takingoff, climbing,
      enroute, descending, landing};
    weather_conditions : { clear, wind, fog};
  end Vars
  Actions = {wind,clear,fog,none};
  Protocol:
    weather_conditions=clear: {clear};
    weather_conditions=wind : {wind};
    weather_conditions=fog : {fog};
    Other : {none};
  end Protocol
  Evolution:
    weather_conditions = clear if Action=clear;
    weather_conditions = wind if Action=wind;
    weather_conditions = fog if Action=fog;
    aeroplane1_status=taxiing if aeroplane1_status=atgate;
    aeroplane1_status=takingoff if aeroplane1_status=taxiing;
    aeroplane1_status=climbing if aeroplane1_status=takingoff;
    aeroplane1_status=enroute if aeroplane1_status=climbing;
    aeroplane1_status=descending if aeroplane1_status=enroute;
    aeroplane1_status=landing if aeroplane1_status=descending;
  end Evolution
end Agent

Agent pilot1
  Lobsvars = {aeroplane1_status};
  Vars:
    perceived_weather : {clear,wind,fog};
  end Vars

  Actions = {pushback,takeoff,engageAP,disengageAP,none};

  Protocol:
    perceived_weather=clear and !(Environment.aeroplane1_status=taxiing or
      Environment.aeroplane1_status=atgate) : {engageAP,disengageAP};
    perceived_weather=wind and !(Environment.aeroplane1_status=taxiing or
      Environment.aeroplane1_status=atgate) : {disengageAP};
    perceived_weather=fog and !(Environment.aeroplane1_status=taxiing or
      Environment.aeroplane1_status=atgate) : {engageAP};

    Other : {none};
  end Protocol
  Evolution:
    perceived_weather=clear if (Environment.Action=clear or Environment.Action=wind);
    perceived_weather=fog if (Environment.Action=fog);
    perceived_weather=wind if (Environment.Action=wind);
  end Evolution
end Agent

Agent autopilot1
  Lobsvars = {aeroplane1_status};
  Vars:
    engaged : boolean;
  end Vars

  Actions = {none};
  Protocol:
    Other : {none};
  end Protocol
  Evolution:
    engaged = true if pilot1.Action=engageAP;
    engaged = false if pilot1.Action=disengageAP;
  end Evolution
end Agent

```



```
Evaluation
  windy if Environment.weather_conditions=wind;
  foggy if Environment.weather_conditions=fog;
  APengaged if autopilot1.engaged = true;
  in_landing_mode if (Environment.aeroplane1_status=descending) or
                    (Environment.aeroplane1_status=landing);
end Evaluation

InitStates
  (Environment.aeroplane1_status=atgate) and autopilot1.engaged=false;
end InitStates

Groups
  pilot = {pilot1};
end Groups

Fairness
  -- in_landing_mode;
end Fairness

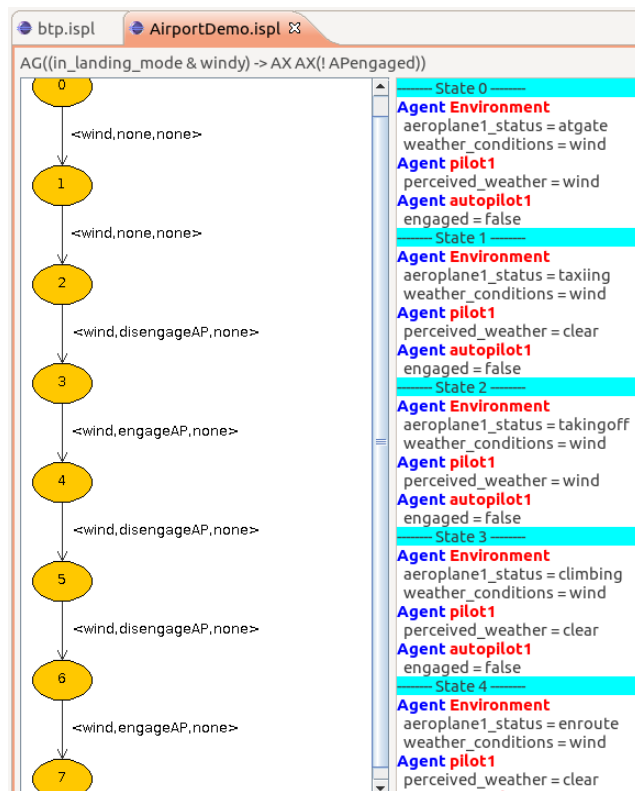
Formulae
  AG((in_landing_mode and foggy) -> AX(AX(APengaged)));
  AG((in_landing_mode and windy) -> AX(AX(!APengaged)));
  <pilot>G(APengaged);
  AG(windy -> K(pilot1,windy));
end Formulae
```

The screenshot shows a window titled 'AirportDemo.ispl' with a 'Verification result' section. It lists four formulas and their corresponding results:

Formula	Result	Action
Formula 1: $AG((in\_landing\_mode \ \&\& \ foggy) \rightarrow AX \ AX \ APengaged)$	TRUE	
Formula 2: $AG((in\_landing\_mode \ \&\& \ windy) \rightarrow AX \ AX \ (! \ APengaged))$	FALSE	show counterexample/witness
Formula 3: $(\langle \text{pilot} \rangle G \ APengaged)$	FALSE	show counterexample/witness
Formula 4: $AG(windy \rightarrow K(\text{pilot1}, windy))$	FALSE	show counterexample/witness

At the bottom right, there is a button labeled 'show BDD information'.

■ **Figure 4** Model checking some formulas on the example scenario.



■ **Figure 5** Counterexample for Formula 2.

## 4.6 Discussion about the testing of MAS

*Working Group 7*

License © Creative Commons BY 3.0 Unported license  
© Working Group 7

### Main questions:

- Commonalities between testing and model checking for MAS ⇒ not discussed
- Do we need special techniques for testing autonomous system?
- What are the testing goals for MAS?

### General considerations:

- Output not deterministically predictable: no “right” or “wrong” of individual decisions.
- Goals and reasoning must be included in the test harness.
- Agents must be able to explain and give rationale for decisions (i.e., no black-box testing is possible)
- Difference to verification? ⇒ considers only a selected number of runs
- Test oracle problem becomes harder (maybe even undecidable)

### Completeness of testing? How to do the testing?

- Test case problem: find a scenario with a well-defined reaction philosophical issue: “highly intelligent” and “defective” cannot be easily distinguished.
- Often “just” random experimentation is done
- Make (random?) experiment and check whether the behaviour can be explained by the overall goals
- Differentiate between high-level goals (on the level of agent groups) and low-level goals (on the level of individual agents)
- Low-level goals are probably easier to check or verify
- Coverage criteria are hard to define.
- Maybe a new coverage criterion. “Strategy coverage” must be defined. Meaning: cover all strategies or cover all strategic decisions? This is different from path coverage (all path with all strategies).

### How to deal with the change of strategies?

- Is a meta-strategy just another strategy?
- Example: cellular-automata game with cats, birds and dogs [(C): Stefan Gruner and his students in Pretoria]
- Can learn strategies
- Evolving strategies are a challenge for MAS
- Testing knowledge and belief versus testing strategies?  
⇒ Probably knowledge and belief are already represented in the strategy.
- Is there a theory of strategy testing?
- Apparently not (yet); e.g. ioco cannot be transferred. Maybe there is a connection to model checking here: evaluation of the quality of testing just by statistical evaluation  
⇒ Simulation environments with random testing coverage of environment vs. coverage of SUT.
- We need a test strategy for testing strategies :-)
- We need suitable criteria for this purpose. One such criterion could be “Robustness”: if the environment changes only little, the agent’s behaviour should typically also change only little.

### What about learning systems?


- MAS vs. von Neumann’s cellular automata

- Programs writing programs?
- Distinguishing a highly intelligent and a defect system can be hard
- Machine learning on two levels:
  - parameter optimization
  - strategy learning
- Learning systems are out of scope at the moment; how about adaptive systems?

OUTLOOK: For the meta theory (Methodology) of testing, can we learn anything from the social sciences or from pedagogics? How do sociologists or school-teachers do their “experiments” with people or pupils? Scientificness of the Methodology must be kept in mind (e.g. Mario Bunge)

## 4.7 Discussion concerning logics for knowledge, time and strategies

*Working Group 5*

License  Creative Commons BY 3.0 Unported license  
© Working Group 5

Logics	Extensions	Limitations	Open questions / Possible solutions
Epistemic Logic	Temporal DEL Memory Synchronous/Asynchronous	Unrealistic w.r.t resources Models are about uncertainty → Awareness? Knowledge de dicto/de re	Alternative semantics
Strategic Logics ATL	Imperfect information Recall Type of strategy Explicit strategies/actions Quantitative aspects	Complexity/Undecidability Where are the “killer” applications? ATL + dynamics Mixed strategies/ probabilistic settings	Security protocols e-voting Plant controller Modelling (translating from semi-formal to formal) More tools Encoding real problems Connection between ATL and game theory
DEL	Connection to LTL + knowledge	Perfect recall unavoidable Asynchronous semantics for DEL	DEL with imperfect recall DEL for planning

## Participants

- Thomas Ågotnes  
University of Bergen, NO
- Carlos Areces  
Universidad Nacional de Córdoba, AR
- Guillaume Aucher  
INRIA Bretagne Atlantique – Rennes, FR
- Alexandru Baltag  
University of Amsterdam, NL
- Ezio Bartocci  
TU Wien, AT
- Ioana Boureanu  
EPFL – Lausanne, CH
- Nils Bulling  
TU Clausthal, DE
- Louise A. Dennis  
University of Liverpool, GB
- Michael Fisher  
University of Liverpool, GB
- Tim French  
The University of Western Australia – Nedlands, AU
- Valentin Goranko  
Technical Univ. of Denmark, DK
- Stefan Gruner  
University of Pretoria, ZA
- Dimitar Guelev  
Bulgarian Academy of Sciences – Sofia, BG
- Yuri Gurevich  
Microsoft Res. – Redmond, US
- Andreas Herzig  
Paul Sabatier University – Toulouse, FR
- Wojtek Jamroga  
University of Luxembourg, LU
- François Laroussinie  
University Paris-Diderot, FR
- Alessio R. Lomuscio  
Imperial College London, GB
- Nicolas Markey  
ENS – Cachan, FR
- Bastien Maubert  
INRIA Bretagne Atlantique – Rennes, FR
- Stephan Merz  
LORIA – Nancy, FR
- Aniello Murano  
University of Napoli, IT
- Wojciech Penczek  
Siedlce University of Natural Sciences and Humanities, PL
- Sylvain Peyronnet  
GREYC – Caen, FR
- Jerzy Pilecki  
Polish Academy of Science, PL
- Sophie Pinchinat  
IRISA – Rennes, FR
- Franco Raimondi  
Middlesex University, GB
- Jean-François Raskin  
Université Libre de Bruxelles, BE
- Markus Roggenbach  
Swansea University, GB
- Ina Schaefer  
TU Braunschweig, DE
- Holger Schlingloff  
HU Berlin, DE
- Gerardo Schneider  
University of Gothenburg, SE
- Henning Schnoor  
Universität Kiel, DE
- François Schwarzentruber  
IRISA – Rennes, FR
- Dmitry Shkatov  
University of the Witwatersrand – Johannesburg, ZA
- Ron van der Meyden  
UNSW – Sydney, AU
- Hans Van Ditmarsch  
LORIA – Nancy, FR
- Ramanathan Venkatesh  
Tata Consultancy Services – Pune, IN
- Karsten Wolf  
Universität Rostock, DE

