# Proceedings of the Automated Reasoning Workshop 2010 Bridging the Gap between Theory and Practice ARW 2010

30th–31st March 2010
University of Westminster
Harrow, United Kingdom

Editor:
Alexander Bolotov

## Program Chair and Organising Chair:

Alexander Bolotov (School of Electronics and Computer Science, University of Westminster, UK)

## Program Commitee

| | |
|---|---|
| Alexander Bolotov | University of Westminster |
| Jacques Fleuriot | Secretary/Treasurer (University of Edinburgh) |
| Simon Colton | (Imperial College London) |
| David Crocker | (Escher Technologies) |
| Louise Dennis | (University of Liverpool) |
| Clare Dixon | (University of Liverpool) |
| Roy Dyckhoff | (University of St Andrews) |
| Ullrich Hustadt | (University of Liverpool) |
| Mateja Jamnik | (Univerity of Cambridge) |
| Alice Miller | (University of Glasgow) |
| Renate Schmidt | (University of Manchester) |
| Volker Sorge | (University of Birmingham) |

## Workshop Website

`http://www2.wmin.ac.uk/bolotoa/ARW/arw-2010.html`

# Preface

This volume contains the proceedings of ARW 2010, the seventeenths Workshop on Automated Reasoning (30th-31st March 2010) hosted by the School of Electronics and Computer Science, University of Westminster, England (UK). Traditionally, this annual workshop which brings together, for a two-day intensive programme, researchers from different areas of automated reasoning, covers both traditional and emerging topics, disseminates achieved results or work in progress. During informal discussions at workshop sessions, the attendees, whether they are established in the Automated Reasoning community or are only at their early stages of their research career, gain invaluable feedback from colleagues. ARW always looks at the ways of strengthening links between academia, industry and government; between theoretical and practical advances. These proceedings contain the abstracts of two invited talks, by Rob Hierons (Brunel University) entitled 'Automated Reasoning and testing' and by Alessio Lomuscio (Imperial College) entitled 'Logic-based specification and verification of multi-agent systems', and nineteen extended abstracts contributed by participants of the workshop.

The abstracts cover a wide range of topics including application of temporal logic in network intrusion analysis, swarm robots verification, and the analysis of subsymbolic sensory information; application of automated reasoning in resolving semantic conflicts and to the analysis of safety and liveness of component-oriented protocols; development of automated reasoning tools for real-time system specification, real arithmetic and experimental mathematics, probabilistic symmetry reduction, study of solvers for propositional dynamic logic, and an instantiation-based theorem prover with equality; analysis of multi-agent verification methods; property preserving generation of large size quasi-groups; terminal complex role inclusion axioms; automatic generation of dynamic investigation problems, reasoning with equality in a contextualised inference system and the analysis of balanced computations through Automated Reasoning.

I would like to thank the members of the ARW Organising Committee for their advice and assistance. I would also like to thank all the colleagues who have helped with the local organisation, namely, Maureen Walker, Jyoti Varsani, Christopher Marston, Rubyna Hague, Peter Cuffari, and Riccardo Piccoli.

London
March 2010

Alexander Bolotov

## Invited Talks

# Contributed Abstracts

# Automated Testing and Reasoning

Robert M. Hierons*

*Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex, UB8 3PH United Kingdom
rob.hierons@brunel.ac.uk

**Abstract**

Software testing and automated reasoning have traditionally concerned different issues and there has been very little interaction between the two communities. However, some connections have been established and especially between automated reasoning and automated test generation. This paper briefly reviews some such relationships.

## 1 Introduction

Testing and reasoning are often seen as opposites: in reasoning we typically try to determine general properties of a model while in testing we observe behaviours of a concrete implementation. There are, however, connections between these areas and in particular if we have a specification or model that has a formal semantics (Hierons et al. (2009)). This paper reviews some connections and points to potential future developments. Its main focus is on the role of automated reasoning in testing and in particular in reasoning about test effectiveness and in driving automated test generation.

## 2 A testing framework

The test framework developed by Gaudel and co-authors (Bouge et al. (1986); Gaudel (1995)) introduced the notion of a *test hypothesis*. A test hypothesis is an assumption, about the system under test (SUT), made by the tester. Given a set of test hypotheses $H$, there may be a finite test suite $T$ that determines correctness relative to $H$: if the SUT satisfies $H$ and passes the tests in $T$ then it must be correct. Such a test suite is sometimes called a *checking experiment*. Given a set of test hypotheses, the process of using testing as part of a proof of correctness can then be divided into the following parts: proving that the SUT actually does satisfy the test hypotheses, potentially through some method other than testing; generating a checking experiment; applying the checking experiment; and comparing the behaviours observed with those in the specification. Three main types of test hypothesis have been identified:

1. Minimality hypotheses, which essentially say that the functionality of the SUT can be expressed in a particular formalism, usually the one used for describing the specification.

2. Uniformity hypotheses, in which we say that certain values are treated in the same way and so the SUT fails on one of these values if and only if it fails on all of these values.

3. Regularity hypotheses, in which we assume that if the SUT is faulty then it will fail with an input containing values with 'complexity' at most $k$ for some predetermined value of $k$ and measure of complexity.

This framework was originally introduced in the context of testing from algebraic specifications. However, it relates to work on testing from a finite state machine (FSM), an area in which the notion of a checking experiment was first described by Moore (Moore (1956)). Here the traditional hypotheses used are: the SUT is equivalent to an (unknown) FSM (a minimality hypothesis); and the SUT is equivalent to an FSM $N$ with at most $m$ states for some predefined $m$ (like a regularity hypothesis).

It is possible to automatically generate a checking sequence given such hypotheses (Chow (1978); Hennie (1964); Hierons and Ural (2006)). However, these approaches apply to deterministic models and implementations. Nondeterminism introduces additional issues since we cannot be certain that all possible responses to a test have been observed unless either the SUT is known to be deterministic or a fairness assumption can be made. If the testers are distributed then there are further issues since we have a very different definition of conformance (Hierons et al. (2008)).

The work on testing from an FSM has been extended to stream X-machines, which include data. It has been shown that FSM based techniques can be applied to stream X-machines as long as additional hypotheses are introduced, these hypotheses being similar to uniformity hypotheses (Holcombe and Ipate (1998).

## 3 Automated test generation

One of the main roles of automated reasoning in testing is test generation. If there is a formal model then sometimes a test generation objective such as executing a particular transition can be represented as a problem to which automated reasoning can be applied (Callahan et al. (1996); Hong et al. (2001)). Typically, a temporal logic property is defined such that a counter-example provides a test that satisfies the test objective. An alternative approach is to use a constraint solver (Pretschner et al. (2005)). Here one typically has to initially symbolically evaluate parts of the model and this can be problematic. Recent work has introduced heuristics that have the potential to produce a path that is *likely* to be feasible; the path can then be symbolically evaluated and test data produced to satisfy the path condition (Kalaji et al. (2009)).

## 4 Conclusions and future directions

While software testing and automated reasoning have traditionally been seen as very different areas, there are now significant connections between them. These mainly concern how automated reasoning can be used to help automate testing. There has been much less work regarding how testing can help automated reasoning but there is potential. For example, we might search for tests that provide counterexamples. Where we wish to apply automated reasoning to a model but are really interested in the properties of an implementation, we might try to learn properties of the SUT through testing and use these properties to refine the model or suggest other properties that may hold.

## References


L. Bouge, N. Choquet, L. Fibourg, and M.-C. Gaudel. Test sets generation from algebraic specifications using logic programming. *Journal of Systems and Software*, 6(4):343–360, 1986.

J. Callahan, F. Schneider, and S. Easterbrook. Automated software testing using model-checking. In *SPIN '96*, pages 118–127. Rutgers Univ., 1996.

T. S. Chow. Testing software design modelled by finite state machines. *IEEE Transactions on Software Engineering*, 4: 178–187, 1978.

M. C. Gaudel. Testing can be formal too. In *Theory and Practice of Software Development (TAPSOFT'95)*, volume 915 of *Lecture Notes in Computer Science*, pages 82–96. Springer, 1995.

F. C. Hennie. Fault-detecting experiments for sequential circuits. In *Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 95–110, Princeton, New Jersey, November 1964.

R. M. Hierons and H. Ural. Optimizing the length of checking sequences. *IEEE Transactions on Computers*, 55(5): 618–629, 2006.

R. M. Hierons, M. G. Merayo, and M. Núñez. Controllable test cases for the distributed test architecture. In *6th International Symposium on Automated Technology for Verification and Analysis (ATVA 2008)*, volume 5311 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2008.

R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. A. Vilkomir, M. R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2), 2009.

M. Holcombe and F. Ipate. *Correct Systems: Building a Business Process Solution*. Springer-Verlag, 1998.

H.S. Hong, I. Lee, O. Sokolsky, and S.D. Cha. Automatic test generation from Statecharts using model checking. In *FATES '01*, volume NS-01-4 of *BRICS Notes Series*, pages 15–30. BRICS, 2001.

Abdul Salam Kalaji, Robert M. Hierons, and Stephen Swift. Generating feasible transition paths for testing from an extended finite state machine (efsm). In *Second International Conference on Software Testing Verification and Validation (ICST)*, pages 230–239, 2009.

E. P. Moore. Gedanken-experiments. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner. One evaluation of model-based testing and its automation. In *27th International Conference on Software Engineering (ICSE 2005)*, pages 392–401. ACM, 2005.

# Logic-based specification and verification of multi-agent systems

Alessio Lomuscio
Department of Computing
Imperial College London
London, UK
A.Lomuscio@imperial.ac.uk

This presentation reports some of our recent work on validation of multi-agent systems. Firstly, basic syntax and semantics of specification languages based on temporal-epistemic logic are discussed. Secondly, labelling algorithms to be used for the verification of systems are introduced together with their efficient implementation via ordered-binary decision diagrams. Thirdly, a demonstration of MCMAS, an open-source model checker for the verification of multi-agent systems is given. Lastly, the talk briefly discusses applications of the methodology in the areas of security protocol analysis, web-services, and fault-tolerant systems.

The talk covers material from (RL05; LQR09; BCL09; LQS08; EL09) and related papers.

## References

[BCL09] I. Boureanu, M. Cohen, and A. Lomuscio. A compilation method for the verification of temporal-epistemic properties of cryptographic protocols. *Journal of Applied Non-Classical Logics*, 19(4):463–487, 2009.

[EL09] J. Ezekiel and A. Lomuscio. An automated approach to verifying diagnosability in multi-agent systems. In *Proceedings of the 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM09)*, pages 51–60. IEEE Computer Society, 2009.

[LQR09] A. Lomuscio, H. Qu, and F. Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In Ahmed Bouajjani and Oded Maler, editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, 2009.

[LQS08] A. Lomuscio, H. Qu, and M. Solanki. Towards verifying contract regulated service composition. In *ICWS*, pages 254–261. IEEE Computer Society, 2008.

[RL05] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, Vol 5(2):235–251, 2005.

# A Network Intrusion Detection System Using Temporal Logic and Stream Processing

Abdulbasit Ahmed, Clare Dixon, and Alexei Lisitsa
Department of Computer Science
University of Liverpool, Ashton Building, Ashton Street Liverpool, L69 3BX , United Kingdom
{Aahmad,CLDixon,Lisitsa}@liverpool.ac.uk

**Abstract**

Intrusion Detection Systems (IDS) aim to detect the actions that attempt to compromise confidentiality, availability, and integrity of a resource by monitoring the events occuring in computer systems and networks. Stream data processing is a database technology applied to flows of data. Temporal Logic is a formalism for representing change over time. In this work, we combine temporal formalisms for representing protocols and attack patterns with stream processing for intruder detection.

## 1 Introduction

Intrusion Detection Systems (IDS) are needed to detect actions that threaten confidentiality, availability, and integrity of resources. *IDS* aim to detect intruders who legitimately pass through the firewall. Moreover, intruders who are inside the firewall (internal users) can only be detected using *IDS* (Stallings, 2000). There are two main types of *IDS* : Host-based Intrusion Detection System (HIDS) in which the *IDS* reside on a single host and monitor all the events for suspicious activity. The other type is Network-based Intrusion Detection System (NIDS) which reside on the network, and is designed to monitor network traffic. The *NIDS* examines the traffic packet by packet in real time, or close to real time, to attempt to detect intrusion patterns (Scarfore and Mell, 2007). There are two main approaches in devising *IDS*: anomaly and misuse based detection systems. In anomaly based detection methods, intrusions are identified as unusual behaviour that differ from the normal behaviour of the monitored system (Denning and Neumann, 1985). The second approach for designing intrusion detection systems is misuse based detection. Attack patterns or signatures are identified and represented in such a way that the system can match these patterns with the log files or network traffic (Lin et al., 1998). Our proposed system is a Network based *IDS* which is capable of detecting intrusions with misuse and anomaly based methods.

## 2 Temporal Logic and IDS

Temporal Logic (TL) is the extension of classical logic with operators that deal with time. TL is an obvious choice for representing misuse signatures or normal behaviour as they often involve temporal events. TL has been used for IDS in the system developed by Naldurg et al. (2004) and is based on *Eagle* (Barringer et al., 2004). *Eagle* is a runtime verification or runtime monitoring system that can use many forms of logic including future and past time temporal logic. Another developed intrusion system based on TL is *Orchids* (Olivain and Goubault-Larrecq, 2005).

In this method, the mechanism of detecting that a signature matches against a sequence of events can be viewed as checking whether a formula representing the signature is satisfied in a model (M). These temporal models are created from a linear sequence of events (logs or packets). So, typically, the idea in misuse based detection is to check that $\phi$ (an attack specification) holds in $M$ ($M \models \phi$). In anomaly based detection, the idea is that the TL Models (the incoming events) are checked against the protocol specification $\phi_n$ rather than a signature. When an event fails to satisfy the specification of the protocol an alert for anomalous behaviour is raised (i.e. $M \not\models \phi_n$).

## 3 Data Stream Processing

Golab and Özsu (2003) defined a data stream as "a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamps) sequence of items". The fundamental difference between traditional database systems and Data Stream Management Systems (DSMS) is that the data takes the form of continuous data streams rather than finite stored data sets. This difference makes DSMS suitable for data-intensive applications where the data model is transient data streams and not persistent relations. These applications could be capital market, network traffic monitoring, telecommunication

data management, and others. DSMS are designed to handle large volumes of data arriving in rapid, time-varying and continuous streams. They can handle queries that are issued once and then continuously evaluated over the data (continuous queries). For example, "write an alert whenever price of X is less than 200". Another useful feature is the sliding window query processing. This sliding window can be based on an ordered field (e.g time) or tuple count. With this feature, recently arrived data is maintained, meaning that old data must be removed as time goes on. It is an approximation technique for bounded memory that we can use to extract a finite relation from an infinite stream and to compute on-line statistics. These features are well suited for applications like network monitoring, network traffic analysis, and intrusion detection.

# 4 Our Research

We are proposing to build NIDS using stream processing technology and temporal logic. This will be a runtime verification system where the data stream represents the model and either the attack pattern or the protocol specification represents the property to be checked. The input to the system, the data stream, is the IP packets (headers and payloads). Attack patterns and simple protocol specifications will be represented in TL. The TL semantics of the attack patterns $\phi$ or protocol specifications $\phi_n$ will be translated into the stream query language for checking if $\phi$ (an attack specification) holds in $M$ ($M \models \phi$) or if the protocol specification $\phi_n$ is not satisfied in $M$ ($M \not\models \phi_n$). The main objectives of our work can be summarized as follows.

**Misuse Attacks** Use stream base processing techniques and features in handling large volume of data in online mode. TL will be used to represent known attacks and then translated into the stream base language.

**Anomaly Attacks** Use stream base processing technique for protocol anomaly detection. TL will be used to write parts of the protocol specification and detect anomalous deviations from this specification.

**Statistical Attacks** Use stream base processing techniques and features in handling large volume of data in online mode to identify attacks based on statistics obtained from the traffic. Currently, we are working on the the first part and we finished from developing a system to handle simple known attacks (attacks caused by a single packet) and the work is in progress to handle more complex temporal attacks. The initial results were encouraging.

# Acknowledgements

We thanks the Stream Base Inc. for their kindness in giving us the necessary license to use their products in our work.

# References

H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-based runtime verification. In *Proceedings of the VMCAI04, 5th International Conference on Verification, Model Checking and Abstract interpretation*, pages 44–57, Venice, Italy, 2004.

D. E. Denning and P. G. Neumann. Requirements and model for IDES: A real-time Intrusion Detection System. Technical report, Computer Science Laboratory, SRI International, 1985.

L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, June 2003.

J. Lin, Wang X. S., and Jajodia. Abstraction-Based Misuse Detection: High-level specification and adaptable strategies. In *Proceedings of the 11th Computer Security Foundation Workshop*, pages 190–201, 1998.

P. Naldurg, K. Sen, and P. Thati. A Temporal Logic Based Framework for Intrusion Detection. In *Proceedings of the 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, Madrid, Spain, 2004.

J. Olivain and J. Goubault-Larrecq. The ORCHIDS intrusion detection tool. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV05)*, 2005.

K. Scarfore and P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). Technical report, National Institute of Standards and Technology, 2007.

W. Stallings. *Network Security Essentials: Applications and Standards*. Prentice Hall, 2000.

# Invariant-Free Deduction for CTL$^\star$: The Tableau Method [‖]

Alexander Bolotov[⋆] [⋆] University of Westminster, HA1 3TP-London, UK.

Jose Gaintzarain[†] [†] The University of the Basque Country, 48012-Bilbao, Spain.

Paqui Lucio[‡] [‡] The University of the Basque Country, 20080-San Sebastián, Spain.

**Abstract**

We define a classical style, one-pass tableau for the full branching-time logic CTL$^\star$. This work extends previously defined tableau technique for propositional linear-time temporal logic, PLTL, giving a new decision procedure for CTL$^\star$. One of the core features of this method is that unlike any other known deduction for the full branching-time logic, it does not require any additional structures to deal with eventualities. Consequently the presented tableau method opens prospect for defining a dual sequent calculus which is cut-free and, in particular, invariant-free. We also hope that this tableau method could serve as a first-step towards an invariant-free resolution method for CTL$^\star$.

*Keywords:* Temporal Logic, Branching-time Temporal Logic, Full Computation Tree Logic, One-pass Tableaux, Invariant-Free Temporal Deduction.

## 1 Introduction

Temporal logic is important in many areas of computer science providing an ideal tool for specifying and modeling the behavior of various classes of dynamic systems, such as reactive systems, digital circuits, concurrent programs, etc. When there is a need for a model for a non-deterministic behaviour, with many possible futures, branching-time logics become essential. Here, most of the research has concentrated on the development of the specification framework given by the Computation Tree Logic (CTL) (Clarke and Emerson (1981)) and a number of its extensions. These extensions are due to the various syntactic restrictions and they culminate in so called Full Computation Tree Logic, CTL$^\star$ (Emerson and Sistla (1984)) which does not have any restrictions on the way how the temporal formulae are assembled.

Developing deductive techniques for branching-time logics means at the same time defining the verification techniques for the corresponding specifications. Referring an interested reader to (Reynolds and Dixon (2005)) and specifically, to (Gore (1999)), for the survey of tableau methods, the subject of this paper, here, we only note that the traditional tableau methods for temporal logic, like the one used in (Emerson and Sistla (1984)), generate auxiliary graphs for some given input which are subsequently checked and possibly pruned. In this paper, we introduce a tableau system for the Full Computation Tree Logic (CTL$^\star$) in a different way. We provide a Branching Tableau Method (BTM) which does not require auxiliary graphs to decide if a set of well-formed CTL$^\star$-formulae is satisfiable. We define a one-pass tableau system avoiding the construction of auxiliary graph and this second, pruning, phase, and thus giving a new decision procedure for CTL$^\star$. We give the algorithm for the systematic construction of the tableau for any set of CTL$^\star$ formulae and established the correctness of these developments. We are not aware of any other one-pass tableau construction for CTL$^\star$. For this expressive logic, which has huge applications, the repository of deductive methods is currently restricted to only a few, quite recently defined, formalisms, both due to Reynolds, namely the axiomatics (Reynolds (2001)) and the tableau (Reynolds (2009)). However, we should note that both of these developments are quite complex and we can see an obvious benefit of our system, first of all, in the transparency of the rules, their intuitive clearness, in providing a method that efficiently limits the length of branches –a problem that stayed open in (Reynolds (2009)) – and, finally, in the algorithm of the systematic tableau construction. Of course, the algorithm itself may be subject to a further investigation into its refinement and this would form part of our future work as well as its implementation.

The results presented in this paper open very good prospect for the development of a corresponding dual cut-free sequent calculus for CTL$^\star$ in line with (Gaintzarain et al. (2009)). We can also see a basis for defining an invariant-free resolution method for the full computation tree logic extending the results in (Gaintzarain et al (2007)) and complementing the existing clausal resolution method for CTL-type branching time logics, initially set up in (Bolotov and Fisher (1999)). Finally, noting that in this paper we have not mentioned anything about the complexity issues, relevant complexity study applied to the tableau algorithm and its refinements will also constitute part of our future work.

[‖]This work has been partially supported by the Spanish Project TIN2007-66523 and the Basque Project LoRea GIU07/35.

# References


Alexander Bolotov and Michael Fisher. A clausal resolution method for CTL branching-time temporal logic. *J. Exp. Theor. Artif. Intell.*, 11(1):77–93, 1999.

E. M. Clarke and E. A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logic of Programs. Proceedings of Workshop*, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer-Verlag, 1981.

E. A. Emerson and A. P. Sistla. Deciding full branching time logic. In *STOC 1984, Proceedings of*, pages 14–24, 1984.

J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. A cut-free and invariant-free sequent calculus for PLTL. In J. Duparc and T. A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2007. ISBN 978-3-540-74914-1.

Jose Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual systems of tableaux and sequents for PLTL. *Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009.

R. Gore. *Tableau methods for modal and temporal logics*, pages 297–396. Kluwer Academic Publishers, 1999.

M. Reynolds and C. Dixon. Theorem-proving for discrete temporal logic. In *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 279–314, 2005.

Mark Reynolds. An axiomatization of full computation tree logic. *J. Symb. Log.*, 66(3):1011–1057, 2001.

Mark Reynolds. A tableau for CTL$^\star$. In Ana Cavalcanti and Dennis Dams, editors, *FM*, volume 5850 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2009.

# Verification of Swarm Robots: the Alpha Algorithm

Clare Dixon*, Alan Winfield† and Michael Fisher*

*Department of Computer Science,
University of Liverpool,
Liverpool, L69 3BZ, UK
{CLDixon,MFisher}@liverpool.ac.uk

†Bristol Robotics Laboratory,
University of the West of England,
Bristol, BS16 1QD
Alan.Winfield@uwe.ac.uk

**Abstract**

Here we apply model checking to a particular swarm robot algorithm, known as the *alpha* algorithm.

## 1 Introduction

A robot swarm is a collection of simple (and usually identical) robots working together to carry out some task Bonabeau et al. (2001); Beni (2005); Sahin and Winfield (2008). Each robot has a relatively small set of behaviours and is typically able to interact with other (nearby) robots and with its environment. Robot swarms are particularly appealing in comparison to one or two more complex robots, in that it may be possible to design a swarm so that the failure of some of the robots will not jeopardize the overall mission, i.e. the swarm is *fault tolerant*. Such swarms are also advantageous from a financial point of view since each robot is very simple and mass production can significantly reduce the fabrication costs.

Despite the advantages of deploying swarms in practice, it is non-trivial for designers to formulate individual robot behaviours so that the emergent behaviour of the swarm as a whole is guaranteed to achieve the task of the swarm, and that the swarm does not exhibit any other, undesirable, behaviours Spears et al. (2004). Specifically, it is often difficult to predict the overall behaviour of the swarm just given the local robot control algorithms. This is, of course, essential if swarm designers are to be able to effectively and confidently develop reliable swarms.

Currently, the analysis of swarm behaviour is typically carried out by experimenting with real robot swarms or by simulating robot swarms and testing various scenarios (eg see Støy (2001); Nembrini (2005)). In both these cases any errors found will only be relevant to the particular scenarios constructed; neither provides a comprehensive analysis of the swarm behaviour in a wide range of possible circumstances. Specifically, neither approach can detect a problem where undesirable behaviour occurs in some untested situation.

In this work we develop the use of temporal verification via model checking for robot swarms in an effort to formally verify that such swarms do indeed exhibit the required global behaviour.

## 2 The Alpha Algorithm

As a case study we consider algorithms for robot swarms which make use of local wireless connectivity information alone to achieve swarm aggregation. Specifically, we examine the simplest (alpha) algorithm described in Nembrini (2005); Winfield et al. (2008). Each robot has range-limited wireless communication which, for simplicity, we model as covering a finite distance in all directions from the location of the robot. Beyond this boundary, robots are out of detection range.

The basic alpha algorithm is very simple:

- The default behaviour of a robot is forward motion.
- While moving each robot periodically sends an "Are you there?" message. It will receive "Yes, I am here" messages only from those robots that are in range, namely its neighbours.
- If the number of a robot's neighbours should fall below the threshold $\alpha$ then it assumes it is moving *out* of the swarm and will execute a $180^\circ$ turn.
- When the number of neighbours rises above $\alpha$ (when the swarm is regained) the robot then executes a random turn. This is to avoid the swarm simply collapsing in on itself.

Thus, we assume that each robot has three basic behaviours: move forward (default); avoidance (triggered by the collision sensor); and coherence (triggered by the number of neighbours falling below $\alpha$). Avoidance is dealt with as follows. If a robot is moving in some direction and the square ahead is occupied: move to the right or left; if these are both occupied move backwards; else stay in the current position. The original direction the robot was moving in is maintained.

# 3 Model Checking Swarm Robots

We apply model checking to prove properties of swarms of robots following the alpha algorithm. In particular we use the NuSMV Cimatti et al. (2002) model checker. To be able to apply model checking we must make the input model discrete and finite. To do this we make the following assumptions:-

- the robot arena is divided into a finite grid of $n \times n$ squares and a robot will move one grid step up, down, left or right according to the alpha algorithm;
- that the robots can move in one of four directions (North, South, East, West) rather than allowing any bearing;
- the grid wraps round, i.e. moving off the right (respectively left) hand side of the grid the robot will reappear at the left (respectively right) hand side and similarly with the top and bottom.

We represent the movement of each robot as a transition system and consider different methods of concurrency, for example synchrony, strict turn taking, non-strict turn taking or fair asynchrony.

Initially the robots may have any direction but are placed on the grid where they are connected but in different grid squares. Initially either robot may move first. We set the value of $\alpha = 1$ i.e. a robot is connected if it can detect at least one other robot. At first we assume that the wireless range is one square in all directions. The property we aim to verify, is that for all robots $i$, $\Box\Diamond con_i$, i.e. each robot stays connected infinitely often.

# 4 Results and Conclusions

We have applied model checking to the alpha algorithm considering differing modes of concurrency and increasing the wireless range. For the two robot case we can show that the required property holds for all grid sizes we tried when we adopt strict turn taking and a wireless range greater than step size. For full details see Dixon et al. (2010). Due to the number of states generated we have only been able to consider small swarms of two or three robots.

Model checking seems to be a useful way of verifying properties of robots swarms allowing us to improve the design of swarm algorithms and guarantee their reliability. We have further work to do to improve our abstraction of the alpha algorithm to make it more realistic. Further, due to the well known state explosion problem we need improved representations to deal with bigger swarms.

# References


G. Beni. From Swarm Intelligence to Swarm Robotics. In *Proc. International Workshop on Swarm Robotics (SAB), Revised Selected Papers*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2005. ISBN 3-540-24296-1.

E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. *Journal of Artificial Societies and Social Simulation*, 4(1), 2001.

A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of International Conference on Computer-Aided Verification (CAV)*, 2002.

C. Dixon, A.F.T. Winfield, and M. Fisher. Temporal Verification of Emergent Behaviours in Swarm Robotic Systems. Submitted, 2010.

J. Nembrini. *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots*. PhD thesis, University of the West of England, 2005.

E. Sahin and A. F. T. Winfield. Special issue on Swarm Robotics. *Swarm Intelligence*, 2(2-4):69–72, 2008.

W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, Physics-Based Control of Swarms of Vehicles. *Autonomous Robots*, 17(2-3):137–162, 2004.

K. Støy. Using situated communication in distributed autonomous mobile robotics. In *SCAI '01: Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, pages 44–52. IOS Press, 2001.

A. Winfield, W. Liu, J. Nembrini, and A. Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2-4):241–266, 2008.

# Towards symbolic reasoning from subsymbolic sensory information

Gaurav Gupta
*School of Electronics and Computer Science
University of Westminster
gaurav.genisys@gmail.com

Alexander Bolotov
†School of Electronics and Computer Science
University of Westminster
a.bolotov@westminster.ac.uk

Alexandra Psarrou
‡School of Electronics and Computer Science
University of Westminster
psarroa@westminster.ac.uk

**Abstract**

This work-in-progress attempts to establish a framework under which partially processed sensory information is represented within a formal system of logic. The goal is to identify possible learning mechanisms using atomic sensory data as a starting point. This approach of applying a formal logical system towards learning is motivated by ability of logical propositions to represent any fact or property describing the world. Additionally, formal logic systems are extremely well developed, with many years of extensive research in various types of logic, and with established proof searching mechanisms. We will test our notion that proof search enables such a system to generate its own hypotheses and then proof-search through the data collected about the world in order to verify or dismiss the hypotheses. We will explore the hypothesis generation process as either a synthetic or an analytical process, or some combination of both. We will attempt to identify effective hypotheses generation mechanisms such that they are goal-directed and reliant on real world information.

Existing learning mechanisms range from application restricted mathematical methods, such as the old architecture of trained neural networks, for example Carpenter and Grossberg (1988), or the alternative method of support vector machines, for example Tong and Chang (2001), to high volume heuristics based methods, such as expert systems with their knowledge acquisition bottlenecks, see Gardiner and McMillan (1990). There has been some research into concept abstraction mechanisms, however the existing approaches face severe limitations. The anchoring method in Zucker et al. (2002) relies on prior information provided in the form of labeled images. It also treats the whole image as an input space, from sets of which it must learn conceptual patterns without knowing what distinct components make up the picture. The conceptual space based method A. Chella and Saffiotti (2004) for anchoring establishes time dependent links between symbols and conceptual data, the anchors approximating only a single generalised object description at every point along the concept space trajectory. Additionally the method applies contraints to trajectories using a priori knowledge. Other existing methods face similar limitations.

The most common type of sensory input is vision. With fairly robust image analysis mechanisms available today, for instance see Comaniciu and Meer (2002), Gupta et al. (2009) and Belongie et al. (2002), extracting meaningful regions from live sensory data is possible. Our goal is to develop an unsupervised concept generation and analysis mechanism that is reliant only on sensory (in this case visual) observations about the environment, with no assumptions or a priori knowledge about what is being observed. Moreover, we do not aim to "anchor" concepts. The term anchoring describes a cognitive bias causing a single characteristic to heavily influence the relative perception of two or more objects. We attempt to automatically generate symbolic representations of sensory information such that any set of percept data captures the greatest detail in individual influences of perceived similarity between real world objects, events or even properties.

We treat different regions within images as distinct sources of information separate from the entire image. The various properties of each region are the observed data. Our approach is a set of learning rules that, when applied to observation based propositional data, allows a rudimentary identification of patterns and associations not only within the observed data but also with respect to the state of the system, which is some measure of the well being of the system. By choosing to monitor desciptions of the world in very physical abstracted terms we raise the chances that important characteristics of the scene are captured in essence. The challenge then is to ensure the set of monitored physical metrics are at least representative of the desired capability of the system. Sets of metrics capable of capturing a particular type of property of one world will do so equally well in all other worlds in which the physical laws do not change.

Any given scenario is composed of interacting parts. We can establish propositional statements describing the component parts, their properties, their behaviour, and most importantly relative changes in each of these. Watching a scene unfold allows us to build up a set of propositional descriptions about what is happening in atomic terms. We may choose to observe n properties for every object. Thus for a simple scene with 3 interacting objects we get a set of 3*n descriptions about the world. However, when we start considering inter-object properties, even simple scenes result in a minimum of 3*n + n! descriptions. For large n, this set holds a staggering number of descriptions. However, every element in this set may not be directly useful for solving problems in specific environments. One of our questions is how do we select an computationally optimal subset of descriptions without sacrificing information resolution? The answer is dependent on

the mechanism we decide to use in order to process the descriptions for some useful outcome. Certain schemes will place more importance on specific kinds of inter-object associations.

If we design the system to specialise in specific circumstances, the next question then becomes how is this different in principle from other learning frameworks that use training phases to adapt to the problem at hand? This issue is addressed by the algorithm applied to select the specialisation pathway. This algorithm considers all properties equal but fades the relevance of those less correlated to the useful outcome and boosts the importance of those taking part directly in the outcome. When the system is applied unmodified to a different scene, other properties are boosted in with the existing ones being faded out if unused.

Once we begin capturing properties about the world we can start generating hypotheses about correlations between the internal state measure and descriptions. Proof searching through collected data validates or negates each hypothesis. Validated hypotheses are the generated goals, and are logical sentences. Components of hypotheses, if linked to an effector response, trigger action. The presentation will discuss the mechanics of the framework and some initial results using synthetic as well as real data.

# References


M. Frixione A. Chella, S. Coradeschi and A. Saffiotti. Perceptual anchoring via conceptual spaces. In *Proceedings of the AAAI-04 Workshop on Anchoring Symbols to Sensor Data*. AAAI Press, 2004.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.993558.

Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.33.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.1000236.

Christopher J. Gardiner and William W. McMillan. Requirements for machine learning in expert systems (abstract). In *CSC '90: Proceedings of the 1990 ACM annual conference on Cooperation*, page 443, New York, NY, USA, 1990. ACM. ISBN 0-89791-348-5. doi: http://doi.acm.org/10.1145/100348.100479.

Gaurav Gupta, Alexandra Psarrou, and Anastasia Angelopoulou. Generic colour image segmentation via multi-stage region merging. *Image Analysis for Multimedia Interactive Services, International Workshop on*, 0:185–188, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/WIAMIS.2009.5031464.

Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM. ISBN 1-58113-394-4. doi: http://doi.acm.org/10.1145/500141.500159.

Jean-Daniel Zucker, Nicolas Bredeche, and Lorenza Saitta. Abstracting visual percepts to learn concepts. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, pages 256–273, London, UK, 2002. Springer-Verlag. ISBN 3-540-43941-2.

# A Comparison of Solvers for Propositional Dynamic Logic

Ullrich Hustadt *Department of Computer Science, University of Liverpool, UK
U.Hustadt@liverpool.ac.uk

Renate A. Schmidt †School of Computer Science, The University of Manchester, UK
Renate.Schmidt@manchester.ac.uk

Propositional dynamic logic (PDL) is an expressive logic for reasoning about programs and actions (Fischer and Ladner, 1979). Initially intended for program verification, it has found applications in a wide range of areas such as verification of rule-based expert systems, synthesis of composite web services, and the formalisation of multi-agent systems. Decision procedures for the satisfiability problem for PDL were first presented by Fischer and Ladner (1979) and Pratt (1980). The satisfiability problem for PDL is EXPTIME-complete. Already the decision procedure of Pratt (1980) was complexity optimal.

In recent years there has been renewed interest in PDL and, in particular, in implementations of theorem provers for PDL. The Tableau Workbench (TWB) includes an implementation of a double exponential time tableau-based algorithm for PDL by Abate, Goré, and Widmann (2009). pdlProver is an implementation of a tableau-based algorithm by Goré and Widmann (2009) that uses caching to achieve complexity-optimality. MLSOLVER by Friedmann and Lange (2009) is a platform for satisfiability checking for various modal fixpoint logics, including PDL. Given a formula it generates a parity game as a product of a tableau for the formula and a deterministic automaton recognising 'bad branches' in the tableau. The parity game is then solved by the PGSOLVER system. LoTREC 2.0 (Gasquet et al., 2005) is a generic tableau-based system for building models of formulae in modal and description logics. It includes a module for PDL, however, it cannot be used as a 'black-box' solver like the other systems and is consequently not included in our comparison.

To compare the TWB, pdlProver and MLSOLVER, we have used two classes of benchmark formulae originally introduced for propositional linear time temporal logic (PLTL) in Hustadt and Schmidt (2002), but reformulated for PDL.

The first class, $\mathcal{C}^1_{\mathrm{PDL}}$, consists of formulae of the form

$$\begin{aligned}&[a^*]\langle a\rangle\top \wedge [a^*]([a]L^1_1 \vee \ldots \vee [a]L^1_k) \wedge \ldots \wedge [a^*]([a]L^l_1 \vee \ldots \vee [a]L^l_k)\\ &\wedge [a^*](\neg p_1 \vee \langle a^*\rangle p_2) \wedge [a^*](\neg p_2 \vee \langle a^*\rangle p_3) \wedge \ldots \wedge [a^*](\neg p_n \vee \langle a^*\rangle p_1),\end{aligned}$$

while the second class, $\mathcal{C}^2_{\mathrm{PDL}}$, consists of formulae of the form

$$\begin{aligned}&[a^*]\langle a\rangle\top \wedge (r_1 \vee L^1_1 \vee \ldots \vee L^1_k) \wedge \ldots \wedge (r_1 \vee L^l_1 \vee \ldots \vee L^l_k) \wedge (\neg r_1 \vee q_1)\\ &\wedge (\neg r_1 \vee \neg q_n) \wedge [a^*](\neg r_n \vee [a] r_1) \wedge [a^*](\neg r_{n-1} \vee [a] r_n) \wedge \ldots \wedge [a^*](\neg r_1 \vee [a] r_2)\\ &\wedge [a^*](\neg r_n \vee [a]\neg q_n) \wedge \ldots \wedge [a^*](\neg r_1 \vee [a]\neg q_n) \wedge [a^*](\neg q_1 \vee \langle a^*\rangle s_2)\\ &\wedge [a^*](\neg s_2 \vee q_2 \vee [a] q_n \vee \ldots \vee [a] q_3) \wedge \ldots \wedge [a^*](\neg q_{n-1} \vee \langle a^*\rangle s_n) \wedge [a^*](\neg s_n \vee q_n).\end{aligned}$$

The $L^i_1, \ldots, L^i_k$ are propositional literals generated by choosing $k$ distinct variables randomly from a set $\{p_1, \ldots, p_n\}$ of $n$ propositional variables and by determining the polarity of each literal with probability $p$. The remainder of each formula only depends on the parameter $n$. In all experiments, for both classes, the parameters $k$, $n$ and $p$ were fixed to 3, 5, and $0.5$, respectively. For each of the values of $l$ between 1 and $8n$ we have generated a test set of 100 formulae, which were tested for satisfiability. Similar to random $k$SAT formulae, formulae in $\mathcal{C}^1_{\mathrm{PDL}}$ and $\mathcal{C}^2_{\mathrm{PDL}}$ are likely to be satisfiable if the number $l$ is small and likely to be unsatisfiable if $l$ is large.

Most of the observations made in (Hustadt and Schmidt, 2002) about the corresponding PLTL formulae carry over to their PDL counterparts. For example, if a formula in $\mathcal{C}^1_{\mathrm{PDL}}$ is satisfiable, then it is satisfiable in a possible worlds model with just $n$ worlds. If a formula in $\mathcal{C}^2_{\mathrm{PDL}}$ is satisfiable, then it is satisfiable in a model with just one world and $r_1$ has to be false at that world. Given these model-theoretic insights about the formulae, their satisfiability is relatively easy to check, in particular, they are as easy to solve as propositional $k$SAT formulae over $n$ propositional variables. But the classes are constructed in such a way that PDL satisfiability checkers, which have to rely on proof-theoretic means, find them challenging. In the case of $\mathcal{C}^1_{\mathrm{PDL}}$, each formula $\varphi_1$ in it imposes a uniform set of constraints on all worlds of a model which gives little guidance in the search for a satisfying model. Furthermore, if the propositional formula $(L^1_1 \vee \ldots \vee L^1_k) \wedge \ldots \wedge (L^l_1 \vee \ldots \vee L^l_k)$ is satisfiable, then potentially every sequence of satisfying truth assignments for this formula could be a model $\mathcal{M}_1$ of $\varphi_1$. Only when we check whether all eventualities $\langle a^*\rangle p_i$ are satisfied within $\mathcal{M}_1$ will we know that our search for a model has been successful. We thus expect that naive tableau-based systems and systems, which like Pratt's method only perform an eventuality check after some exhaustive search for candidate models, will perform poorly. The class $\mathcal{C}^2_{\mathrm{PDL}}$ is meant to illustrate how quickly a tableau-based system can find a model for a formula provided it makes the right choices for disjunctive formulae and how efficiently it can recover from making the wrong choices.

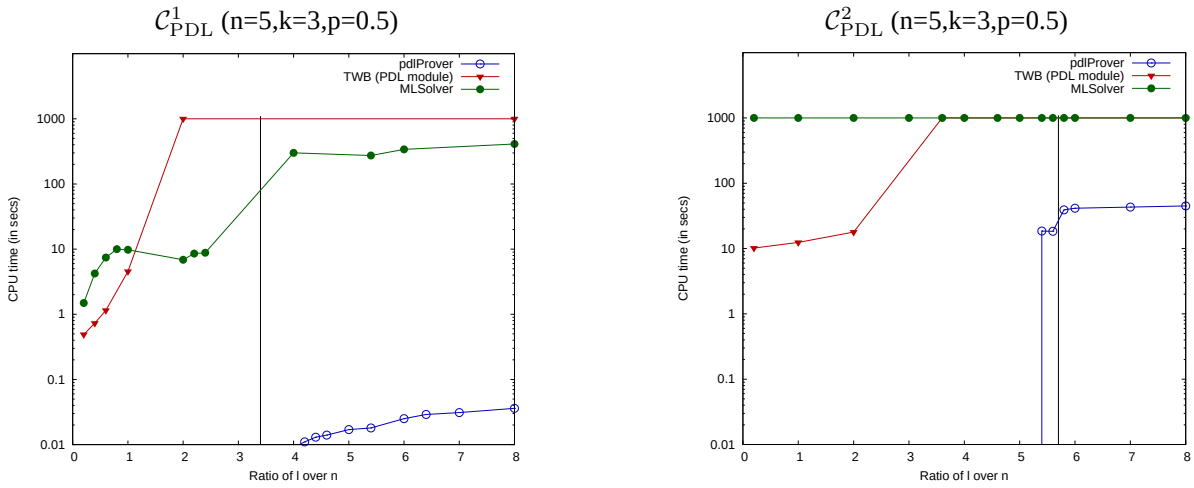



Figure 1: Performance of the decision procedures

Figure 1 shows the median CPU time graphs for all three procedures on $\mathcal{C}^1_{\mathrm{PDL}}$ and $\mathcal{C}^2_{\mathrm{PDL}}$. In each graph the vertical line indicates the ratio $l/n$ at which test sets contain 50% satisfiable and 50% unsatisfiable formulae. The tests have been performed on PCs with Intel Core 2 Duo E6400 CPU @ 2.13GHz with 3GB main memory using Fedora 11. For each individual satisfiability test a time-limit of 1000 CPU seconds was used.

As can be seen in Figure 1, $\mathcal{C}^1_{\mathrm{PDL}}$ separates pdlProver, the only system which uses caching, from the other two. Caching allows a prover to take advantage of the uniformity of the constraints imposed on the worlds of a model by formulae in $\mathcal{C}^1_{\mathrm{PDL}}$. Thus, the good performance of pdlProver on this class was predictable. The absence of similar optimisations in the PDL module of the Tableau Workbench and in MLSOLVER are the most likely explanation for their poor performance. However, even then one might have expected both systems to be able to solve formulae in $\mathcal{C}^1_{\mathrm{PDL}}$ with $l/n > 6$, which are almost all unsatisfiable and have a very constrained and limited search space for models.

For $\mathcal{C}^2_{\mathrm{PDL}}$ the ideal system has negligible median runtime for $l/n < 5.7$, as up to this point the majority of formulae is satisfiable and a model for a satisfiable formula can easily be found. Only pdlProver could be 'guided' to behave in the expected way and to make the right choices in the model construction up to $l/n \leq 5.4$ which is almost 'optimal' (by inputting formulae in the 'right' form). In contrast, the PDL module of the Tableau Workbench and MLSOLVER fail to show a similar behaviour. For MLSOLVER we also observe a marked difference between $\mathcal{C}^1_{\mathrm{PDL}}$ and $\mathcal{C}^2_{\mathrm{PDL}}$. While for $\mathcal{C}^1_{\mathrm{PDL}}$ MLSOLVER was able to solve the majority of formulae for each ratio $l/n$, on $\mathcal{C}^2_{\mathrm{PDL}}$ the opposite is true and it solved not a single formula in this class.

Overall, pdlProver shows the best performance on these two classes of PDL formulae. The experiments illustrate the importance of caching for PDL systems, but also show that there is still considerable room for improvement. In addition, the experiments show that the two classes of benchmark formulae originally devised for PLTL are also useful for 'black-box' performance evaluations of PDL solvers.

# References


P. Abate, R. Goré, and F. Widmann. An on-the-fly tableau-based decision procedure for PDL-satisfiability. *Electr. Notes Theor. Comput. Sci.*, 231:191–209, 2009.

M. J. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. Comp. and System Sci.*, 18:194–211, 1979.

O. Friedmann and M. Lange. A solver for modal fixpoint logics. In *Prelim. Proc. M4M-6*, pages 176–187. Roskilde University, Denmark, 2009.

O. Gasquet, A. Herzig, D. Longin, and M. Sahade. LoTREC: Logical tableaux research engineering companion. In *Proc. TABLEAUX'05*, volume 3702 of *LNAI*, pages 318–322. Springer, 2005.

R. Goré and F. Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Proc. CADE-22*, volume 5663 of *LNCS*, pages 437–452, 2009.

U. Hustadt and R. A. Schmidt. Scientific benchmarking with temporal logic decision procedures. In *Proc. KR2002*, pages 533–544. Morgan Kaufmann, 2002.

V. R. Pratt. A near-optiomal method for reasoning about actions. *J. Comp. and System Sci.*, 20:231–254, 1980.

# Automated Reasoning in Resolving Semantic Conflicts across Heterogeneous Repositories

Pavandeep Kataria
University of Westminster
P.Kataria1@wmin.ac.uk

Radmila Juric
University of Westminster
Juricr@wmin.ac.uk

**Abstract**

Semantically related data across heterogeneous data repositories may have a number of semantic similarities because they model real life concepts which may have identical or overlapping meaning. These semantic similarities may trigger a variety of semantic conflicts due to the differences in either the interpretation of semantically related data in respect to their meaning in a given context, or the intended use of semantically related data within a given context, or the way we have modeled semantically related data in a universe of disclosure. Semantic conflicts, if not resolved, may become an obstacle for achieving interoperability across heterogeneous data repositories, which is particularly important if we know that semantically related data exist across them.

## 1 Introduction

We propose an automated reasoning mechanism which resolves semantic conflicts though various ontological layering created by ontological mappings: ontology alignment, ontology integration and ontology merge. Thus ontology alignment, integration and merge of ontological concepts are performed though a set of reasoning rules which in turn resolve various semantic conflicts at different ontological layers.

Ontological reasoning mechanisms are ideal for capturing the content of heterogeneous repositories in terms of understanding their semantically related data and structuring the semantic conflicts contained within them. Therefore, the power of ontological models and the manipulations of their semantics allow the modelling of ontological mappings between two or more semantically related concepts according to the type of conflict between them.

Using reasoning rules as an extension of ontological expressivity we resolve semantic conflicts by defining a set of restrictions / conditions that describe their level of equivalence in terms of their degree of similarity. Furthermore, we use rule chaining to allow the automation of inference / assertion in terms of transforming semantically related ontological concepts into their equivalent states.

### 1.1 Automation of Reasoning

We have achieved a high level of automation in terms of manipulating the semantics stored in ontologies and inferring ontological individuals, thus performing reasoning rules chaining through the software application. We use semantic web technology, Protégé editing tool, OWL ontologies, SWRL reasoning rules and JESS engine. The software application, developed with Net beans, which accommodates the reasoning mechanism, automates semantic retrievals across heterogeneous repositories, which are placed within a pervasive healthcare domain.

Ontological reasoning is performed through (i) exploiting OWL 2.0 based ontological constructs (which includes the manipulation of ‘object’ properties) and (ii) triggering ‘rule chaining’ across ontological layering in terms of running SWRL rules upon ontological concepts using the Pellet reasoning engine.

Figure 1 illustrates a software architecture based on Generic Ontology for Context Aware, Interoperable and Data Centric (Go-CID) software applications (Kataria et al. 2009), which shows the environment where automated reasoning has been performed. Software applications, requirements upon them and ontologies with their mapping through reasoning rules are shown in layers and described in our previous publications (Kataria and Juric 2010), (Kataria and Juric 2009) and (Kataria et al. 2008)

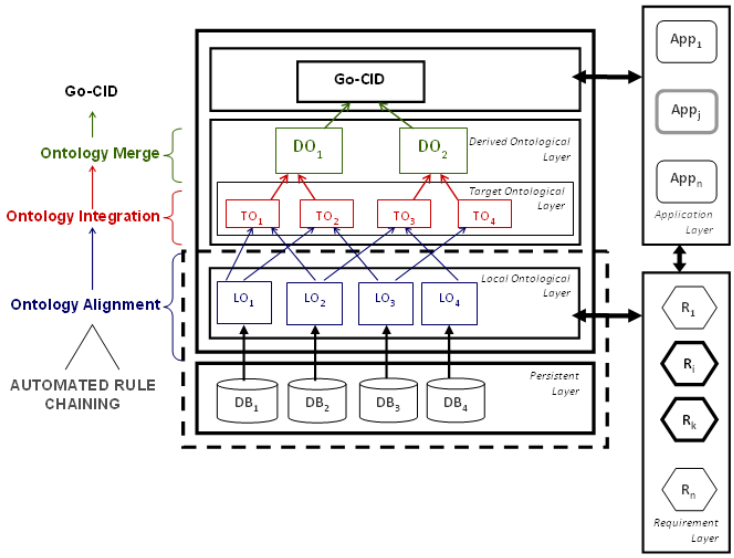



Figure 1: A software architecture based on Generic Ontology for Context Aware, Interoperable and Data Centric (Go-CID) software applications and automated reasoning performed through ontology mappings.

# References


Kataria Pavandeep, Macfie Alex, Juric Radmila, and Madani Kambiz. Ontology for Supporting Context Aware Applications for the Intelligent Hospital Ward. *Transaction of Integrated Design & Process Science Tran-Disciplinary International Journal*, 12(3): 35-44, 2008

Kataria Pavandeep, Ganguly Sukanta, Juric Radmila, Atila Ertas and Murat Tanik (2009). Sharing Information Across Heterogeneous e-Health Systems. *Telemedicine and e-Health Journal*, 15(5): 454-464, 2009.

Kataria Pavandeep, and Juric Radmila. Sharing Healthcare Data by Manipulating Ontological Individuals. *Twelfth Transdisciplinary Conference-Workshop on Integrated Design & Process Science Tran-Disciplinary Workshop*, CD-ROM 2009.

Kataria Pavandeep and Juric Radmila. Sharing Healthcare Data through Ontological Layering. *43rd Annual Hawaii International Conference on System Sciences (HICSS 43)*, pp.1-10, 2010.

# Model Checking Multi-Agent Systems

Ryan Kirwan
*Department Of Computing Science
University Of Glasgow
r.kirwan.1@research.gla.ac.uk

Alice Miller
†Department Of Computing Science
University Of Glasgow
alice@dcs.gla.ac.uk

**Abstract**

Model checking (2) has been applied to many areas of software and hardware verification; this includes hybrid systems, which contain both hardware and software components. To model hybrid systems both types of component are included in a single model. In this research we consider hybrid systems in which multiple agents interact in an environment. An agent is a simple robot capable of moving, detecting obstacles, and learning to avoid obstacles. The goal of this research is to discover a practical and standardised way of modelling these systems, and to develop theoretical techniques to measure performance.

## 1 Introduction

Model checking involves verifying properties of a system and it can be an important tool for system development. The importance of model checking is due to the impracticality of running exhaustive testing to verify properties of real physical systems. Even when exhaustive testing can be used to verify a system's properties, model checking could be used to verify the same properties faster and more accurately, through automated exhaustive checking on abstracted system models. The goal of this research is to avoid system testing by using model checking to apply verifications to multi-agent systems.

The multi-agent systems we are modelling contain two or more identical robots that are learning to avoid each other and obstacles in a walled environment. These robots have 2 long range "distal" sensors and 2 short range "proximal" sensors; distal sensors are used to avoid collisions, proximal sensors are used to detect collisions.

### 1.1 Initial Model Checking

To better understand the problem domain, the modelling of existing multi-agent systems was undertaken. The aim was to use mainstream model checkers such as PRISM (4) and SPIN (5). These model checkers were chosen to model systems used in experiments performed at the University of Glasgow (7). We work closely with the developers of these experiments; this allows us access to new and interesting systems and allows our results to influence their designs.

The explicit state model checker SPIN is well established in the field of model checking, and allows us to verify properties of systems described in the model specific language Promela (5). Promela is an intuitive language which allows models to be created quickly and easily. PRISM is a symbolic model checker which allows it to verify much larger state-spaces than SPIN. It also has the benefit of quantitative analysis and probabilistic weightings. The quantitative analysis can provide a performance measure for multi-agent systems, which SPIN cannot. However, PRISM has a less comprehensive modelling language than Promela.

In our models each robot is treated as a process, as is the environment. The environment is aware of the position of all things within it. The robots are able to communicate to the environment via sensors. The environment is abstracted to a grid and the robots can move in either of the 8 compass directions [1], moving 1 cell at a time. When a robot choses an adjacent cell to move to, it first uses its sensors to check that the grid cell is vacant.

The physical dimensions of the robots are created to an accurate scale based on the resolution of the environment's grid. The environment is represented as a grid with resolution of 22x22 cells, the outer cells form a surrounding wall, and each robot is 2x2 cells with distal sensor antenna 4 grid cells long and proximal antenna 1 grid cell long.

Instead of static, cell-to-cell movements, the model is designed to better emulate the continuous driving of real robots, by making small adjustments as they move. Each robot reassesses its direction every time it arrives at a new grid position and since the resolution of the grid is relatively high each calculation for a new movement is done at small intervals, hence closely emulating the continuous reassessment of the real system.

The calculations for the robots' movement are based on the difference between the Manhattan distances (1) of each of the robots' distal antenna sensors. Distal sensors are long rods protruding at 45 degrees from the robot and are used for

[1] In reality it would be possible to measure the direction a robot is facing to exact degrees of accuracy, but having 8 directions allows the robots to access all adjacent grid cells. This creates an accurate model without overloading the number of directions in which a robot can face.

avoiding, whilst the proximal sensors are impact sensors which are close to the robot. The distal sensors send signals of varying strength depending on how close or far away an obstacle is to the robot. This movement calculation emulates the avoidance behaviour of the real robots.

### 1.2 Results

Modelling with SPIN allows properties to be checked, such as: is it possible for "Robot1" to collide with "Robot2", or other obstacles? This property can be expressed in Linear Time Temporal Logic (LTL) (6), thus:

**[] ( ! ( ((robo1X==robo2X) && (robo1Y==robo2Y)) || ((robo1X==obVal) && (robo1Y==obVal)) ) )**

Modelling with PRISM allows us to check Probabilistic Computation Tree Logic (PCTL) (6) properties such as, what is the probability of an erroneous state being reached? Synchronizing on their movements, the Steady-state probability of the system having two agents occupying the same grid cell is 0.0064. This was verified using the PCTL property:

**S=? [(robo1X==robo2X) & (robo1Y==robo2Y)]**

To make the PRISM model accurate each robot must be able to move in and out of synchronization with each other robot. To allow this the PRISM code must specify the probability of the agents synchronizing and not. There is not an accurate way of selecting this probability, yet the value chosen for it will affect every subsequent validations' probability.

These types of property are useful, but are based on assumptions. Assumptions are made so models can be abstracted enough to apply verification. Examples of these assumptions are that sensors were assumed to behave like perfect springs whenever an obstacle or agent touched a point on one of the sensor antenna. It was assumed that the robots moved and turned with consistent accuracy. Another assumption is that the discrepancies in the positioning of the sensors relative to the robots' facing direction was insignificant —this discrepancy arose from abstracting the environment to a grid. The relevance of the verification becomes unclear once these assumptions are factored in.

### 1.3 Future Work

We plan to use Hybrid Modelling Languages such as Hytech (3), to create highly detailed system models. We will also continue working closely with current experiments involving multi-agent systems in order to expand our knowledge of how to model such systems. Principally we want to develop various models, classifying the benefits and costs of each approach. One further aim is to develop a custom-made tool to model these types of multi-agent system.

## References


[1] Paul E. Black. Manhattan distance. *Dictionary of Algorithms and Data Structures, NIST*, 2009.

[2] Jr. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, February 2000.

[3] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: a model checker for hybrid systems. 1997.

[4] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. *LNCS*, 3920, 2006.

[5] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Pearson Education, 2004.

[6] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2 edition, 2004.

[7] P. D. Prodi, B. Porr, and F. Worgotter. Adaptive communication promotes sub-system formation in a multi agent system. *(private communication)*.

# Reasoning in Pervasive Computational Spaces

Nigel Koay
University of Westminster
N.Koay@my.wmin.ac.uk

Preya Syal
University of Westminster
Preya.Syal@my.wmin.ac.uk

Radmila Juric
University of Westminster
Juricr@wmin.ac.uk

**Abstract**

We briefly describe the process based on automated reasoning which enables us to create pervasive computational environments across problem domains. Our idea is to model the semantics of pervasive spaces in ontologies and perform ontological reasoning in order to determine which types of services we want and which preferences we impose on the environment when requesting such services.

## 1 Introduction

We create a pervasive ENVIRONMENT suitable for "smart homes" and model it as a set of "safety and security" SERVICES in terms of "detecting leakage of carbon monoxide, checking if windows are open and spotting unknown person on premises" etc. We also have preferences of PEOPLE living in "smart homes" in terms of specifying which devices would deliver such SERVICES: "CCTV cameras should be visible from the outside smart homes and all devices within the smart home must be wireless". Our reasoning mechanism should give us a list of devices which form a special aspect of the pervasive environment and satisfy our preferences at the same.

The same reasoning mechanism may be performed in a "smart classroom" where SERVICES are related to lectures, their topics and learning outcomes; ENVIRONMENT is equivalent to a particular learning environment, its class, teaching material and technology needed for the class and PEOPLE are students with their backgrounds (courses and degrees) and preferences (availability; time of lecture etc.) Our reasoning mechanism should give us a list of the best teaching materials which are essential in delivering a lecture of a given topic and according to student preferences.

## 2 The Reasoning Process

The process in Figure 1 consists of 3 steps which actually correspond to three types of ontological reasoning: Step 1 is illustrated with the 'circled 1a & 1b' and Step 2 by the 'circled 2a & 2b'. They perform matching between ENVIRONMENT and PEOPLE and between ENVIRONMENT and SERVICES respectively. In our process, the ontological matching is performed through reasoning rules thus:

(I) Rule 1 is used to match the characteristics of ENVIRONMENT with the personal preferences of PEOPLE (a user) who might be using the ENVIRONMENT

(II) Rule 2 is used to match a purpose of the ENVIRONMENT with the personal preferences of PEOPLE (user) who might be using the ENVIRONMENT.

(III) Rule 3 identifies the best possible ENVIRONMENT (Pervasive Computing Environment) because it matches its purpose and characteristics with SERVICEs requested by PEOPLE, which take into account the PEOPLE's preferences.

Rule 1, 2 and 3 can be automated in terms of running them through a software application built upon RULE_3 result set (which are ontological concepts). We have implemented our process in a few problem domains (Koay et al, 2010a), (Syal and Juric, 2010), (Koay et al, 2010b), (Ayim et al, 2009), (Koay et al, 2009).

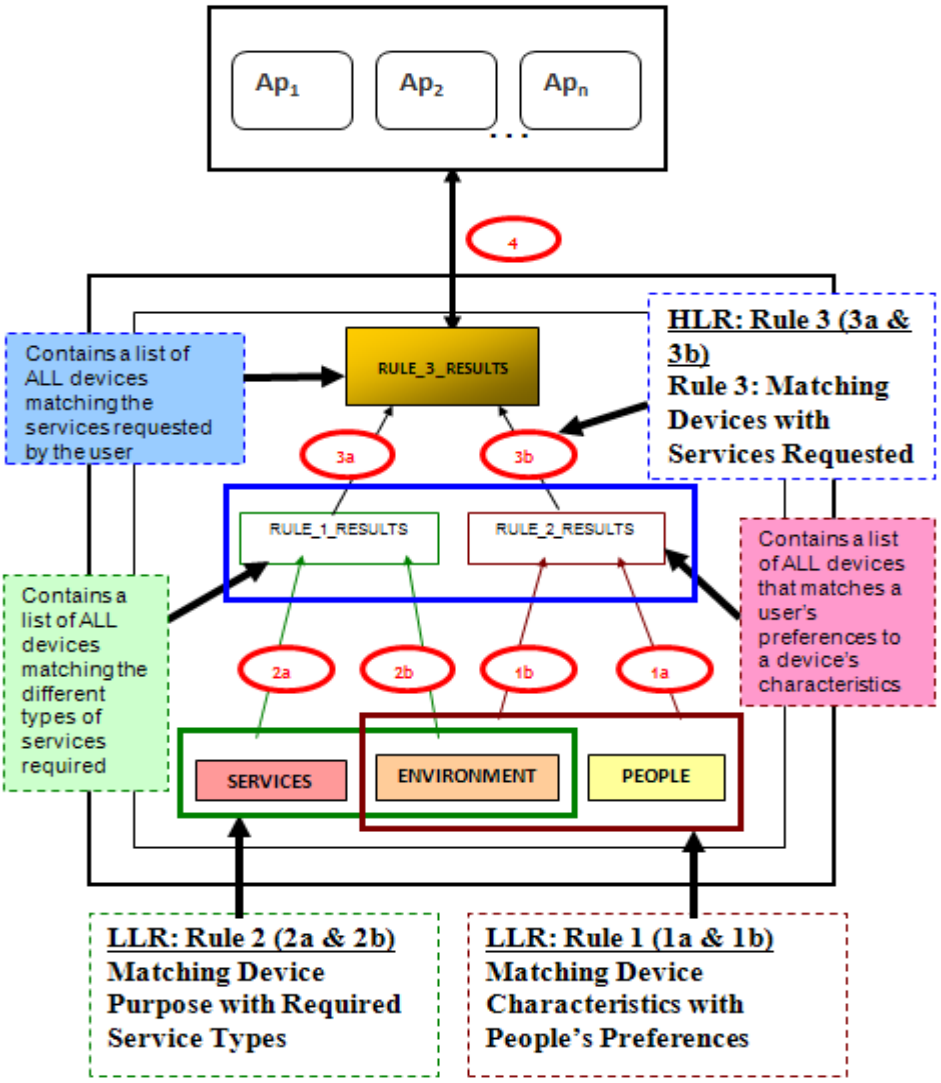



Figure 1: Reasoning Process for Creating Pervasive Computational Spaces

## References


Abena Ayim, Sukanta Ganguly, Nigel Koay, Radmila Juric. Semantic Management of Privacy Policies in Social Network Sites. In *Proceedings of the 12th International Conference on SDPS 2009*, University of Auburn at Montgomery, Al, November 1-5 2009.

Nigel Koay, Radmila Juric, Braimah Lat. PERCEON – The Pervasive Computing Environment Ontology for Smart Homes. To appear in *Proceedings of the 15th International Conference on System Design and Process Science*, SDPS 2010, June 2010, Dallas, US

Nigel Koay, Pavandeep Kataria, Radmila Juric, Gabor Terstyansky, Patricia Oberndorf. Ontological Support for Managing Non-Functional Requirements in Pervasive Healthcare. In *Proceedings of the 42nd Hawaii International Conference on System Science 2009*, HICSS 42, Hawaii, Big Island, US, January 5-8, 2009.

Nigel Koay, Pavandeep Kataria, Radmila Juric. Semantic Management of Non-Functional Requirements in e-Health Systems. To appear (accepted) in *The Tele-Medicine and e-Health Journal*, ISSN 1530-5627, 2010.

Preya Syal, Radmila Juric. Defining Smart Learning Environments through Ontological Reasoning. To appear in *Proceedings of the 15th International Conference on System Design and Process Science*, SDPS 2010, June 2010 Dallas, US

# iProver-Eq: An Instantiation-Based Theorem Prover with Equality

Konstantin Korovin*
*University of Manchester
korovin@cs.man.ac.uk

Christoph Sticksel†
†University of Manchester
csticksel@cs.man.ac.uk

## 1 Introduction

Instantiation-based methods are a class of deduction calculi for first-order clausal logic. The common idea is to instantiate clauses and to employ efficient propositional or more general ground reasoning methods in order to prove unsatisfiability or to find a model. Among other important properties, Instantiation-based methods naturally decide the first-order logic fragment of effectively propositional logic (EPR) which has interesting applications (see, e.g., Baumgartner (2007) for an overview).

iProver-Eq is an implementation of an instantiation-based calculus Inst-Gen-Eq which is complete for first-order logic with equality and decides the EPR fragment modulo equality. The system is an extension of the successful iProver system and preserves the characteristic feature of combining first-order reasoning with efficient ground satisfiability checking where the latter is delegated in a modular way to any state-of-the-art SMT solver, i.e. a solver for ground satisfiability modulo theories.

In the following we outline the iProver-Eq system and present its calculus for equational reasoning to generate instances.

## 2 System Overview

The basic idea of the Inst-Gen method, introduced in Ganzinger and Korovin (2003), is as follows. The set of first-order clauses is abstracted to a set of ground clauses by mapping all variables to the same ground term. An SMT solver is harnessed to check if the ground abstraction of the clauses is unsatisfiable, in which case the set of first-order clauses is also unsatisfiable. Otherwise, there is a ground model for the abstraction that is used to guide an instantiation process. The model is represented as a set of abstracted literals and an attempt is made to extend it to a model of the first-order clauses by reasoning on the first-order literals corresponding to the abstracted literals in the model. When this fails, new (not necessarily ground) instances of clauses are generated in a way that forces the ground solver to refine the model in the next iteration.

The saturation process in iProver-Eq is outlined in Figure 1. Two major components there are unit superposition for equational reasoning on literals and an SMT solver for ground reasoning, which are both non-trivial processes. Unit superposition will be described in the next section. The ground solver is regarded as a black box.

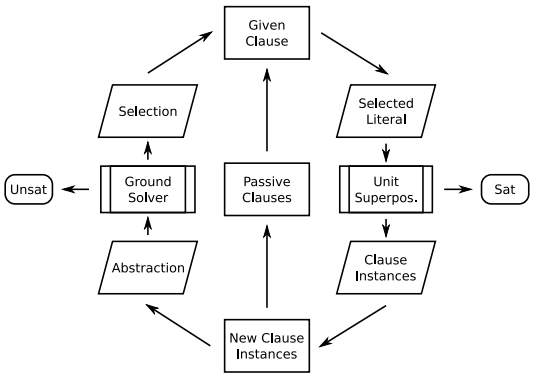



Figure 1: Saturation process in the iProver-Eq system

The saturation process is based on a given clause algorithm, which partitions the set of clauses into two disjoint sets, namely the Inst-Active and the Inst-Passive clauses. Initially, there are no Inst-Active clauses, all input clauses are considered to be new instances, their ground abstractions are passed to the ground solver which is then invoked to either return a model of the abstraction or to conclude its unsatisfiability. The new clauses are moved to the Inst-Passive set from where in each step of the process a clause, called the given clause, is chosen and put into the Inst-Active set. Using the current model of the ground abstraction, one of the literals in the given clause is selected and passed to the unit superposition calculus. If a subset of the selected literals is found to be inconsistent by the unit superposition calculus, then corresponding instances of clauses are added to the set of new clauses. The process continues by adding the abstractions of the new clauses to the SMT solver, running the solver on the extended set of ground clauses and moving the new clauses to the Inst-Passive set. The process maintains the invariant that the ground abstractions of the selected literals in the set of Inst-Active clauses are consistent and have been passed to the unit superposition component. iProver-Eq terminates with a result of unsatisfiable if the ground solver reports an unsatisfiable abstraction. If the Inst-Passive clause set is empty and the selected literals are consistent as stated by the unit superposition component, iProver-Eq terminates with the result satisfiable.

## 3 The Unit Superposition Calculus

If the set of selected (not necessarily ground) literals is consistent, a model for the set of Inst-Active clauses ex-

**Selection**

$$\frac{}{\{C\} \mid D\colon L}$$

where $C \in S$ and $\mathrm{sel}(C) = L$ and $D$ is a constraint on clause $C$

**Superposition**

$$\frac{\ell_1 \mid D_1\colon l \simeq r \qquad \ell_2 \mid D_2\colon L[l']}{(\ell_1 \cup \ell_2)\theta \mid (D_1 \wedge D_2)\theta\colon L[r]\theta}\ (\theta)$$

where $L[l'] = u[l'] \simeq v$ or $L[l'] = u[l'] \not\simeq v$ and (i) $\theta = \mathrm{mgu}(l, l')$, (ii) $l'$ is not a variable and (iii) for some grounding substitution $\mu$ (iiia) $l\theta\mu \succ r\theta\mu$, (iiib) $u[l']\theta\mu \succ v\theta\mu$ and (iiic) $\mu$ satisfies the constraint $(D_1 \wedge D_2)\theta$

**Equality Resolution**

$$\frac{\ell \mid D\colon l \not\simeq r}{\ell\colon \square}\ (\theta)$$

where $\theta = \mathrm{mgu}(l, r)$ and satisfies the constraint $D$

Figure 2: Labelled Unit Superposition

ists and it has thus been proved satisfiable. Otherwise, there is an inconsistent subset of the selected literals. The clauses that these literals are selected in are instantiated such that the inconsistency is witnessed by the ground abstraction. For non-equational literals it suffices to search for unifiable complementary literal pairs. However, in the presence of equations, we apply the unit superposition calculus in order to find inconsistent literals and to obtain clause instances.

For simplicity, we only consider pure equational logic where all atoms are equations, different clauses are assumed to be variable-disjoint. The inference rules of the unit superposition calculus are shown in Figure 2 and are similar to the standard superposition calculus, see, e.g., Nieuwenhuis and Rubio (1999).

Each literal in the calculus has a label consisting of a set of clauses and a constraint that is used for redundancy elimination which we will not describe here. When the selected literal of the given clause is received, we label it with the given clause and a constraint. The conclusion of an inference is labelled with the union of the labels of its premises where the unifier $\theta$ has been applied to each clause. The constraint is the conjunction of the constraints of its premises where again the unifier $\theta$ has been applied.

Ganzinger and Korovin (2004) originally defined the unit paramodulation calculus and a way to extract instantiating substitutions from proofs. Our addition of labels to literals replaces their extraction of substitutions. Instead of having to trace a proof tree to the literals at its leaves in order to obtain substitutions to be applied to clauses, our labels directly display the clause instances while still allowing for constraint notions to eliminate redundancy.

The conditions on applicability of an inference are independent of labels which are thus merely an annotation to facilitate the generation of clause instances from sets of inconsistent literals. Therefore, the same literal with different labels has the same conclusions. Further, it is well known from paramodulation approaches, that all variants of a literal allow the same inferences with conclusions that are variants of each other. Obviously, one wants to combine all labels of all variants of a literal in order to avoid duplicating the search for possible inferences for each literal. Although it is possible to merge all labels into one set of clauses, the challenge is to compactly represent combined labels in a way that preserves the constraint notions.

# 4 Conclusion

In general, not all optimisations from standard paramodulation-based calculi can be lifted to the unit superposition calculus. We have to take care not to omit clause instances that are required by the ground solver to witness inconsistency of its model. Nevertheless, our labelled calculus enables simplification by demodulation and there is a powerful semantic notion of redundancy which can be used to justify further techniques.

iProver-Eq makes use of state-of-the-art implementation techniques like term indexing for unification. It is reasonably efficient on the TPTP benchmark although it is in an early stage.

# Acknowledgements

Our work is based on the iProver system, as in described in Korovin (2008) and Korovin and Sticksel (2010). We thank Renate Schmidt for helpful discussions.

# References

Peter Baumgartner. Logical Engineering with Instance-Based Methods. In *CADE-21*, volume 4603 of *LNAI*, pages 404–409. Springer, 2007.

Harald Ganzinger and Konstantin Korovin. New Directions in Instantiation-Based Theorem Proving. In *LICS 2003*, pages 55–64. IEEE, 2003.

Harald Ganzinger and Konstantin Korovin. Integrating Equational Reasoning into Instantiation-Based Theorem Proving. In *CSL 2004*, volume 3210 of *LNCS*, pages 71–84. Springer, 2004.

Konstantin Korovin. iProver - An Instantiation-Based Theorem Prover for First-Order Logic (System Description). In *IJCAR 2008*, volume 5195 of *LNCS*, pages 292–298. Springer, 2008.

Konstantin Korovin and Christoph Sticksel. iProver-eq – An Instantiation-based Theorem Prover with Equality. submitted to IJCAR 2010, 2010.

Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 1999.

# Property Preserving Generation of Large Size Quasigroup-structures

Quratul-ain Mahesar and Volker Sorge
School of Computer Science
University of Birmingham
{Q.Mahesar|V.Sorge}@cs.bham.ac.uk

## 1 Introduction

Quasigroups are simple algebraic structures whose operation has to satisfy only a single axiom, the Latin square property. Quasigroups are generalisation of groups and in general are non-associative. Consequently there exists a very large number of different finite quasigroups already for very small orders. This combinatorial explosion makes them ideal candidates for applications where the generation of a large number of simple structures is necessary such as in cryptography. On the other hand the number and lack of structure makes them difficult to handle algebraically, in particular to enumerate or to classify. The aim of our research is to develop methods to automatically generate quasigroups of large size by bootstrapping structural properties of smaller size quasigroups, instead of exhaustive search. We employ automated discovery techniques for the construction of generating systems that allow an easier computation of large size structures. This construction will be property preserving in order to ensure a goal-directed generation of structures. That is, given a set of properties we want a quasigroup to have, the approach should automatically generate a means to construct a large order quasigroup exhibiting these properties. The work has applications both in the pure mathematical theory to solve open existence problems for finite quasigroups and loops, as well as potentially in areas such as the generation of cryptographically strong quasigroups.

Our work builds on previous work by (Sorge et al., 2008) that was concerned with the generation of classification theorems in quasigroup theory. The procedure incorporated a set of diverse reasoning techniques, including first order resolution theorem proving, model generation, satisfiability solving and computer algebra methods. We intend to further exploit these techniques and in particular the concept of generating system introduced in that work for the goal directed construction of quasigroups.

## 2 Generating Systems: Generators and Relations

The concept of generating systems for quasigroups was introduced in (Sorge et al., 2008) and can be used to determine a quasigroup structure of size $n$ using $n$ complex equations rather than $n^2$ simple equations of its Caley table. We define a word of a quasigroup $Q$ with binary operation $*$ as the combination of elements $a_1, ..., a_n \in Q$ under the operation $*$ and write $w(a_1, ..., a_n)$ for short. The concept of generating systems can then be defined as follows:

**Definition 2.1.** *Let $Q$ be a finite quasigroup with binary operation $*$, and let $q_1, ..., q_n \in Q$ be the elements of $Q$. Let $a_1, ..., a_m \in Q$ where $n, m \in \mathbb{N}$ and $1 \leqslant m \leqslant n$. Then, we define the generating system $G$ for $Q$ as follows:*

$$G = \langle \{a_1, ..., a_m\} | \{q_1 = w_1(a_1, ..., a_m), ..., q_n = w_n(a_1, ..., a_m)\} \rangle$$

- *The set of elements $\{a_1, ..., a_m\} \subseteq Q$ are called the generators.*
- *$\{w_1(a_1, ..., a_m), ..., w_n(a_1, ..., a_m)\}$ represents a set of words. Every element $q \in Q$ can be expressed as a word which is called a relation or factorisation.*

Note, that the generating system for quasigroups is different from the one for groups in that every single element is explicitly defined. For example the quasigroup $Q_1$ on the right is determined by the generating system $G_1 = \langle \{a\} | \{b = a * a, c = (a * a) * (a * a)\} \rangle$.

| $Q_1$ | a | b | c |
|---|---|---|---|
| a | b | a | c |
| b | a | c | b |
| c | c | b | a |

## 3 Using Automated Discovery and Reasoning Techniques

We will use automated discovery and reasoning techniques during the construction of generating systems for larger order quasigroups. In particular, we will employ concept formation to automatically construct extended generating systems for larger order quasigroups and we are currently extending the machine learning system HR (Colton et al., 1999) accordingly. HR uses production rules to transform the input data table of a concept into the resulting data table that represents a novel concept. We are currently experimenting with HR by both exploiting existing production rules and introducing novel,

bespoke ones for extending generating systems. For example, the above quasigroup $Q_1$ has an additional property that it is commutative. If we want to generate an Abelian quasigroup $Q_2$ of order 4, we can extend the generating system $G_1$ to a generating system $G_2$ by adding a further relation:

$$G_2 = \langle \{a\} | \{b = a * a, c = (a * a) * a, c = a * (a * a), d = ((a * a) * a) * ((a * a) * a)\} \rangle$$

| $Q_2$ | a | b | c | d |
|---|---|---|---|---|
| a | b | c | | |
| b | c | | | |
| c | | | d | |
| d | | | | |

$G_2$ fixes four elements in $Q_2$'s Caley table (given on the left) and the remainder of the table can be filled in using the Latin square property, which yields the full table given on the right. $Q_2$ is again commutative, which is obvious from both the Caley table and the construction of $G_2$.

| $Q_2$ | a | b | c | d |
|---|---|---|---|---|
| a | b | c | a | d |
| b | c | d | b | a |
| c | a | b | d | c |
| d | d | a | c | b |

In addition to a production rule adding relations to an existing generating system using the original generators, we can also define one that extends the number of generators employed. During the execution of a production rule, HR already uses theorem proving and model generation to ensure consistency and usefulness of novel concepts. For production rules adding relations and generators we have additional proof problems to consider, for which we will employ theorem provers, model generators and SAT solvers for the following abstract proof problems:

1. Does a new relation indeed yield a new element and not one of the elements already present in the generating system?
2. Will the expanded generating system, indeed preserve the property of the smaller order quasigroup?

We will use symbolic computation techniques to construct the skeleton Caley tables for the quasigroups. Constraint solving techniques will then be used in order to fill the remaining missing elements in the skeleton Caley table of the quasigroups generated by the generating system. We are interesting in using implied constraints that logically follow from the initial model of a constraint satisfaction problem. The addition of implied constraints can greatly improve efficiency and is an important step in solving difficult problems. Constraint solving techniques have been previously used by (Charnley et al., 2006), where the theorems generated by HR are used as constraints to reformulate the basic constraint satisfaction model for finding examples of quasigroups. We intend to further explore this method in our research which would be novel as implied constraints have never been used before for solving problems involving large size quasigroups.

# 4 Open Quasigroup Existence Problems

As one application we will work towards solving some of the open problems in quasigroup theory that are concerned with the existence of quasigroups of a particular size exhibiting particular properties. For example, when considering quasigroup $(Q, *)$ with associated left division $\backslash$, where $a \backslash c = b \Leftrightarrow a * b = c$ for every $a, b, c \in Q$, an operation $\circ$ can be defined on $Q$ by $x \circ y = x * (y \backslash x)$. It is currently an open problem if there exist quasigroups of order $q = 14, 18, 26$ or $42$ such that $(Q, \circ)$ is a quasigroup. For instance, the following example of order 4 shows that the operation $\circ$ does in general not form a quasigroup operation as the Latin square property does not hold:

| * | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 4 |
| 2 | 4 | 1 | 3 | 2 |
| 3 | 3 | 4 | 2 | 1 |
| 4 | 1 | 2 | 4 | 3 |

| $\backslash$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 3 | 1 | 2 | 4 |
| 2 | 2 | 4 | 3 | 1 |
| 3 | 4 | 3 | 1 | 2 |
| 4 | 1 | 2 | 4 | 3 |

| $\circ$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 3 | 4 | 2 |
| 2 | 4 | 2 | 3 | 1 |
| 3 | 4 | 2 | 3 | 1 |
| 4 | 3 | 1 | 2 | 4 |

Previously, some open problems in quasigroup theory have been solved by (Fujita et al., 1993; Zhang and Hsiang, 1994) where model generation techniques were used to solve existence problems for small quasigroups. However, the goal-directed generation of large size quasigroups is still out of the scope of current Automated Reasoning technologies.

# References


J. Charnley, S. Colton, and I. Miguel. Automatic generation of implied constraints. In *Proc. of ECAI 2006 conference*, pages 73–77. IOS Press, Netherlands, 2006.

S. Colton, A. Bundy, and T. Walsh. Automatic concept formation in pure mathematics. In *Proc. of IJCAI'99*, pages 786–791. Morgan Kaufmann, USA, 1999.

M. Fujita, J. Slaney, and F. Bennett. Automatic generation of some results in finite algebra. In *Proc. of IJCAI'93*, pages 52–57. Morgan Kaufmann, USA, 1993.

V. Sorge, S. Colton, R. McCasland, and A. Meier. Classification results in quasigroup and loop theory via a combination of automated reasoning tools. *Commentationes Mathematicae Universitatis Carolinae*, 49(2):319–339, 2008.

H. Zhang and J. Hsiang. Solving open quasigroup problems by propositional reasoning. In *Proc. of International Computer Symp*, 1994.

# Terminal Complex Role Inclusion Axioms

Nenad Krdžavac[⋆†]
[⋆]University of Belgrade, Serbia
nenadkr@tesla.rcub.bg.ac.rs

Milneko Mosurović[†]
[†]University of Montenegro, Montenegro
milenko@ac.me

**Abstract**

The paper defines an extension of complex role inclusion axioms (RIAs) in $\mathcal{SROIQ}$ description logic (DL) with terminal complex RIAs.

## 1 Introduction

Strict partial order on the set of role names in $\mathcal{SROIQ}$ (3) DL, as logical basis for OWL 2 language (1), does not allow cyclic dependencies among role names in RIAs (3). Extending the regularity conditions on the set of roles, Kazakov proved that the OWL 2 ontology with role hierarchy, where RIAs induces regular languages, is decidable (4). A tableau-based procedure for $\mathcal{SROIQ}$ logic does not use syntactic restrictions directly (4). As an input, it uses finite automata for RIAs (3) (4). Some restrictions have been used for automata construction (4). One can use the same tableu-based procedure (3) for any RIAs for which finite non-deterministic automata can be constructed (4).
Let's consider role RIA of the form $PQ \sqsubseteq R$. By the semantics of RIA (3), for every pair $(x, y) \in (PQ)^{\mathcal{I}}$ implies $(x, y) \in R^{\mathcal{I}}$. Again, by the semantics one cannot define other individuals such that $(\exists x_1, y_1)(x_1, y_1) \in (PQ)^{\mathcal{I}} \wedge (x_1, y_1) \notin R^{\mathcal{I}}$.
To address this problem we proposed RIAs of the form $R \equiv \rho_R$ (5), where $\rho_R$ is a regular expression which corresponds to the role name R. We also defined additional restrictions on the set of RIAs to regain decidability of defined logic. We extended the solution (5) by defining terminal roles in complex RIAs. Such solution has next properties:

1. The existing OWL 2 ($\mathcal{SROIQ}$) (3) language is a special case of this language,
2. one can use existing tableau procedure for $\mathcal{SROIQ}$ DL (3) to check the satisfiability of the ontology $\mathcal{O}$ defined in this paper,
3. for the role hierarchy which does not generate regular languages, one can define some roles as terminal. It is useful in case of incomplete reasoning,
4. one can extend this method to define terminal role at an arbitrary "level" (for example level 2).

## 2 Terminal Roles

**Definition 1** *RIA with terminal roles is an expression of the form $R_1R_2 \cdots R_i|R_{i+1} \cdots R_n \sqsubseteq R$, where $R_i|$ denotes that $R_i$ in composition $R_1 \cdots R_iR_{i+1} \cdots R_n$ is a terminal role with respect to role R.*

Intuitively, $R_i$ is a terminal role if it is not a composition of roles with appearing role R in that composition.

**Definition 2** *An array of states $s_0, s_1, \cdots, s_k$ is a path in the model $\mathcal{I}$ for $\rho = R_1 \cdots R_k$ if $(s_{j-1}, s_j) \in R_j^{\mathcal{I}}$. Removing a path for $\rho$ would change the interpretation of a role $R_j$ for $R_j \backslash \{(s_{j-1}, s_j)\}$.*

**Definition 3** *$\mathcal{I}$ is a model for RIA of the form $R_1R_2 \cdots R_i|R_{i+1} \cdots R_n \sqsubseteq R$ if*

1. *$(x, y) \in (R_1R_2 \cdots R_n)^{\mathcal{I}}$,*
2. *There exists a path $x = s_0, s_1, \cdots, s_n = y$ in the model $\mathcal{I}$ for $\rho = R_1 \cdots R_n$ such that for every word of the form $\rho_1 R \rho_2$, $(s_{j-1}, s_j) \notin (\rho_1 R \rho_2)^{\mathcal{I}}$, when in the interpretation $\mathcal{I}$ removes path $s_0, s_1, \cdots, s_n$.*

*Then $(x, y) \in R^{\mathcal{I}}$.*

**Theorem 1** *$\mathcal{SROIQ}$ ontology with terminal roles is undecidable.*

**Proof.** The proof follows from the proof of theorem 6 defined in (2).
In order to have decidable language we introduce restrictions on RIAs, as defined in (3), based on strict partial order. Suppose that $\leq$ is partial order on the set of roles such that, if $R \leq T$ then $R^- \leq T^-$. We denote $R \simeq T$ if $R \leq T$ and $T \leq R$, but if $R \leq T$ and it is not $T \leq R$ then $R < T$.

**Definition 4** *Allowed RIAs (necessary for decidability) with respect to $\leq$ are expressions of the form*

1. $R_1 R_2 \cdots R_n \sqsubseteq R$ *and* $R_i < R$, *for all* $1 \leq i \leq n$,
2. $R R_1 R_2 \cdots R_n \sqsubseteq R$ *and* $R_i < R$, *for all* $1 \leq i \leq n$,
3. $R_1 R_2 \cdots R_n R \sqsubseteq R$ *and* $R_i < R$, *for all* $1 \leq i \leq n$,
4. $R^- \sqsubseteq R$,
5. $RR \sqsubseteq R$,
6. $R_1 R_2 \cdots R_i | R_{i+1} \cdots R_n \sqsubseteq R$ *where* $R_i \simeq R$ *and* $R_j < R$, *for all* $j \neq i$.

Suppose that there exists partial order $\leq$ on the set of roles in the ontology $\mathcal{O}$, such that all RIAs in $\mathcal{O}$ are allowed with respect to $\leq$.

**Definition 5** *Let $\sqsubseteq_{\mathcal{O}}$ be a minimal relation among arrays of roles and arbitrary role names which satisfy the following conditions:*

1. $R \sqsubseteq_{\mathcal{O}} R$
2. $\rho \sqsubseteq_{\mathcal{O}} R \wedge \rho_1 R \rho_2 \sqsubseteq_{\mathcal{O}} R_1 \in \mathcal{O} \rightarrow \rho_1 \rho \rho_1 \sqsubseteq_{\mathcal{O}} R_1$
3. $\rho \sqsubseteq_{\mathcal{O}} R \wedge \rho_1 R | \rho_2 \sqsubseteq_{\mathcal{O}} R_1 \wedge \rho \neq \rho^{'} R_1 \rho^{''} \rightarrow \rho_1 \rho \rho_1 \sqsubseteq_{\mathcal{O}} R_1$

**Definition 6** *Language $\mathcal{L}_{\mathcal{O}}(R) = \{w | w \sqsubseteq_{\mathcal{O}} R\}$ corresponds to role $R$ in ontology $\mathcal{O}$.*

**Theorem 2** *For every role $R$ in $\mathcal{O}$, language $\mathcal{L}_{\mathcal{O}}(R)$ is regular and one can construct finite automaton for $\mathcal{L}_{\mathcal{O}}(R)$.*

*Proof.(Sketch)* One can construct an automaton inductively over the relation $\leq$ as defined in (3). One case, where $R_i \simeq R$, with $R_i$ a terminal role with respect to role $R$, is an exception. The construction of the automaton for a terminal role is independent of the construction of the automaton for role $R$.

**Theorem 3** *Ontology $\mathcal{O}$ is decidable.*

*Proof.(Sketch)* The proof follows from theorem 2 and from the observation that the tableau-based procedure (3) accepts finite automata for given RIAs.

**Example 1** *Suppose that ontology $\mathcal{O}$ has RIAs of the form: $P \sqsubseteq_{\mathcal{O}} P^-; PQ | \sqsubseteq_{\mathcal{O}} R; PR | \sqsubseteq_{\mathcal{O}} Q$. Language $\mathcal{L}_{\mathcal{O}}(R) = \{R, PQ\}$ is finite. In (4) authors considered such kind of RIAs but without terminal roles. Intuitively, new defined semantics of complex roles in this paper (definition 3) ignores multiple repeating of role P in the front of role Q. It means that without terminal roles, any model $\mathcal{I}$ for ontology $\mathcal{O}$ satisfies that $(x, y) \in (PPPQ)^{\mathcal{I}}$ implies that $(x, y) \in R^{\mathcal{I}}$. With terminal roles the last conclusion is not true. Also, without terminal roles the language $\mathcal{L}_{\mathcal{O}}(R)$ is infinite. Note also that such kind of RIAs are not allowed in $\mathcal{SROIQ}$ DL (3).*

# References


[1] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. P. Schneider, and U. Sattler, OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*. 6(4):309−322, 2008.

[2] I. Horrocks and U. Sattler. Decidability of SHIQ with Complex Role Inclusion Axioms. *Artificial Intelligence*, 160(1-2):79-104, December 2004.

[3] I. Horrocks, O. Kutz, and U. Sattler. *The Even More Irresistible $\mathcal{SROIQ}$*. In American Association of Artificial Intelligence Press, editor. In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), 57−67, 2006.

[4] Y. Kazakov. *An Extension of Regularity Conditions for Complex Role Inclusion Axioms*. I. Horrocks, B. Motik and U. Sattler, editors, In Proceedings of the 22nd International Workshop on Description Logics (DL 2009). volume 477, July 2009.

[5] N. Krdžavac, M. Mosurović. *An Extension of Role Inclusion Axioms with Regular Expressions*. In Proceedings of Information Technology -IT 2010, Žabljak, Montenegro, 2010.

# Please Hold While We Try to Connect You

Shamimabi Paurobally*
*School of Electronics and Computer Science
University of Westminster, London W1W 6UW, U.K.
paurobs@wmin.ac.uk

Jim Cunningham†
†Computing Department
Imperial College, London SW7 2BZ, U.K.
rjc@doc.ic.ac.uk

**Abstract**

Agent interaction in realistic applications is subject to many forms of uncertainty – including *information and network uncertainty, trust of and conflicts with other participants, lack of stability in a deal and risks about agreements and commitments.* However, one of the most common forms of uncertainty occurs when a group has divergent beliefs about the interaction they are engaged in – some agents believe an agreement has been reached, while others believe it has been rejected or that they are still bargaining. Such misunderstandings can arise because of loss of network performance, spurious connections, message loss or delays, or the other party is still engaged in its decision making. In our work, we aim to facilitate a group of agents to attain the same beliefs about an interaction, *independent of the reliability of the underlying communication layer*. Previous work on bilateral synchronisation between agents will be extended to consider the possibility of attaining reliable multilateral interactions between more than two distributed agent systems.

## 1 Introduction

Social interactions, such as cooperation, coordination and negotiation, are a fundamental feature of multi-agent systems. They are enacted through a variety of interaction protocols, here regarded as the public rules or norms for communications of the participants of a group when carrying out some social encounter. In this context, the protocol ensures that all group participants following it can expect certain responses from others and can coordinate meaningfully towards a goal. But in many cases it is not always clear what it means for a group of agents to follow a protocol. In particular, issues arise when unexpected events occur, for example agents do not comply with or misunderstand the interaction protocol, or the communication is faulty. To deal with such issues and the semantics of agent communications, we regard an interaction as a joint process between agents. The steps in such a joint process progress by virtue of the propositions believed by the group. For the purpose of reasoning about the beliefs of a group of agents, we consider the state of an interaction as entailed from the propositions believed by all the agents about that interaction and derivable from an interaction protocol. For example, let a common protocol *P*, for a joint negotiation between two agents *X* and *Y*, specify that after agent *X* has browsed a catalogue, it may make an offer which should be followed by an agreement or a rejection from *Y*. A state of negotiation such as *offered* means that both agents believe an action *offer* has been made and on reaching an agreement the state changes from *offered* to *agreed*. In our work, we view common beliefs as relating to the shared beliefs of a group of agents about each other, and joint beliefs as deriving from the union of all the individual beliefs of agents in a group.

In this paper, we give and overview of previous work on synchronisation of interactions between agent systems. In current work, we are considering what should be the assumptions and requirements for reliable multi-lateral interactions between distributed agents to ensure the safe progression of their conversation through a number of identifiable states and termination in a consistent manner.

## 2 Synchronisation Protocols

In Paurobally et al. [2] we propose that agents' beliefs regarding the protocol state could be synchronised by adding a synchronisation layer, which includes protocols that synchronise the agents' beliefs as shown in figure 1. We proved that under certain assumptions, it is possible to achieve reliable interaction between two agents.

### 2.1 Shared Beliefs

We use *belief*, instead of *knowledge*, because the property of *knowledge* being *true* makes it is far harder to attain than belief in a practical context. Whereas knowledge must be true, beliefs only require consistency. For example, an agent *X* may believe an agreement has been reached with agent *Y* when in fact it has misunderstood or mis-implemented the protocol, or there has been a security breach with a malicious agent impersonating *Y*. When these uncertainties are compounded with network unreliability, it is easier for an agent to believe some state than to know that state.

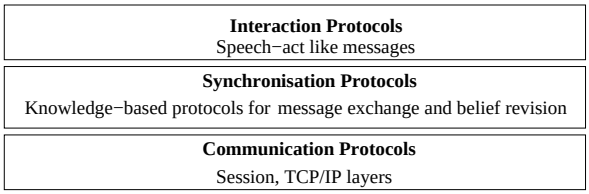



Figure 1: Layers of Protocols

While an interaction protocol may be considered as common belief in a group of agents, the participants' individual beliefs about the progress of a particular interaction are liable to differ between the time a message is sent and received. For example, a sender has added beliefs about the message it sent compared to another agent which has not received that message. Therefore, some degree of common belief, and consistency in joint beliefs about the interaction state, are necessary to safely progress in an interaction and to avoid contradictions and any confusion that may arise from a participant persisting with differing belief about the interaction state. This issue of consistency is an important problem because unreliable communication is the norm in the communication networks and infrastructures in which agents are most likely to be deployed. The key issue here is that the interaction must not continue without ensuring that all agents believe the same state at some point. For example, if messages are lost in an interaction following the protocol *P* introduced above, then one agent may believe the state to be *agreed* while another believes it to be *browsed*. Such discrepancies may lead to disputes worsened when monetary, time or safety-critical information are involved.

We use an axiomatic belief system for consistency and introspection and assume that each agent in a group has such a system of beliefs and is aware that others do.

In [2], we showed that shared beliefs (if everyone believes the state $\alpha$ and believes that everyone believes so too - ($E_G\alpha \wedge E_G^2\alpha$)) can bring about reliable interaction and agreement between two agents, under certain assumptions about an imperfect underlying communication layer.

### 2.2 Joint Conversations

Later work aimed at providing reliable interaction in the case of more than two agent systems, to ensure that all the agents commit to belief update for the progression of an interaction. Thus in [3] , we examined the synchronisation problem from the point of view of committing to a joint conversation. We argued that a proper theoretical treatment of conversations cannot be simply derived compositionally from the semantics of individual Communicative Acts (CAs). Accordingly, we developed a theory of joint conversations that is independent of its constituent CAs by treating the process of a group following an interaction protocol as a persistent joint communicative action (JCA) by the group. We proposed a theory for representing and reasoning about joint conversations, defined compliance in a joint conversation and we proved salient properties of joint conversations.

Dunn-Davies et al. [1] argue that synchronisation between more than two agents is not possible - that multi-lateral synchronisation is an insoluble problem. Nonetheless we are interested in how to counter that argument by investigating what assumptions and requirements are to be made to achieve satisfactory synchronisation in multi-agent systems. First, given that we are using a belief system, it may be viable for an agent to commit itself to a belief update in order to start a belief update revision with the rest of the agents. We also assume that the underlying communication layer e.g. TCP/IP will deal with message duplication, re-ordering, mutation and phantom messages.

## References

[1] H. Dunn-Davies, R. Cunningham, and S. Paurobally. Synchronization protocols for reliable communication in fully distributed agent systems. In *Proc. AAMAS*, 2008.

[2] S. Paurobally, R. Cunningham, and N. R. Jennings. Ensuring consistency in joint beliefs of interacting agents. In *Proc. AAMAS*, pages 662–669, 2003.

[3] S. Paurobally and M. Wooldridge. Joint conversation specification and compliance. In *DALT 2007, Lecture notes in computer science (4897). Springer, Berlin*, pages 18–35, 2007.

# Safety and Liveness of Component-oriented Protocols: A Feasibility Study

Shamima Paurobally, Alexander Bolotov and Vladimir Getov
*School of Electronics and Computer Science
University of Westminster, London W1W 6UW, U.K.
{paurobs,bolotov,V.S.Getov}@wmin.ac.uk

**Abstract**

Interaction in both agent-based and component-based architectures is facilitated by sharable, verified and unambiguous protocols with desirable properties. An interaction protocol may be expressed as a logical theory e.g. dynamic logic, joint intention theory or event calculus, thereby enabling the proof of its properties and its correctness. This paper focuses on new joint work in specifying and proving properties such as safety and liveness, of interaction protocols, in component-based frameworks. A safe and sound interaction allows no unpredictable states and transitions, and states allowed for the behaviour are only those that are defined by the protocol. Other properties such as termination, soundness, completeness, stability and fairness can be specified and verified for such protocols.

## 1 Introduction

Setting up an automated negotiation involves deciding on a common language, ontology and choosing a common negotiation protocol. Since the protocols are shared by participants, their properties have to be validated for stable and fair negotiations. In component-based models and corresponding methodologies [4], the higher level of complexity involves a wider range of requirements and resources, which in turn requires dynamic intelligent properties and flexibility resembling. These similarities make it interesting to bring the study of agent oriented interactions into the component-based framework.

It thus becomes necessary to define and prove safety and liveness properties in such large-scale component-oriented architectures. For example, because of the large number and complexity of the interactions between the components in service composition, we need to ensure a certain stability and fairness in the provision of services in service-oriented distributed frameworks and infrastructures.

## 2 Safety and Liveness Properties in Component-Based Frameworks

Given the sophisticated and complex interactions typically occurring in extreme scale component oriented frameworks, it is desirable to use formalisms with support for concurrency and verification for the service composition.

Safety and liveness properties were first introduced by Lamport [3] and have since then been extensively studied in the verification of concurrent programs. The liveness property would normally assert that a process eventually enters a desirable state and that there are no deadlocks while the safety property would assert that 'nothing bad' happens. There is a need, however, to adapt the standard notions of liveness, termination and soundness to the interaction/ composition protocols. In the component-based framework we would speak about the liveness of the processes involved into this interaction as well as about the liveness of the system as whole.

In order to achieve the required level of abstraction we need to use high level specification languages capable of tackling both the nature of the desired properties and the dynamism of the overall system and its environment. We will utilise the experience of invoking the specification languages of rich branching-time logics CTL and its extensions, in the framework of component-based systems [1]. This will enable the application of the deductive verification techniques for proving properties of the protocols.

In order to prove safety, liveness, and termination, the behaviour of an execution and possible sequences of actions in a component-based protocol are examined. The interaction terminates if it can be proved that all paths in a protocol lead to a terminal state without getting stuck in an infinite loop. A safe and sound interaction allows no unpredictable states and transitions, and states allowed for the behaviour are only those that are defined by the protocol. Liveness and termination properties cover the absence of deadlock and livelock.

However, a corresponding monitoring mechanism, one of the core parts of the proposed architecture which would feed the specification with the required data, should be developed. Indeed, proving that a component-based protocol satisfies

a property may not always be feasible by only monitoring the executions of an interaction for violations of the property. While it is possible to monitor an interaction for violations of safety properties, this is not so easy for liveness properties. In addition, the problem is intensified by the extreme-scale nature of the distributed services computing framework.

Another avenue of future work is to analyse the game theoretic properties of the protocol. These include Pareto efficiency, stability, possible equilibrium, deception-free and conflict resolution [5].

# References


[1] Alessandro Basso, Alexander Bolotov and Vladimir Getov. *Temporal Specification and Deductive Verification of a Distributed Component Model and Its Environment*. In *Proceedings of the International COnference Secure System Integration and Reliability Improvement (SSIRI2009)*, 2009, pp 379-386, IEEE Computer Society, Los Alamitos, CA, USA.

[2] E. A. Emerson. Automated reasoning about reactive systems. In *Logics for Concurrency: Structures Versus Automata, Proc. of International Workshop*, volume 1043 of Lecture Notes in Computer Science, pages 41–101. Springer, 1996.

[3] L. Lamport. Sometimes is sometimes "not never" - on the temporal logic of programs. In *7th ACM Symposium on Principles of Programming Languages*, pages 174–185, 1980.

[4] Kung-Kiu Lau and Zheng Wang. *Software Component Models*. In *IEEE transactions on Software Engineering*, Vol. 33, No. 10, October 2007, pp 709-724.

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

# An Approach to Probabilistic Symmetry Reduction

Christopher Power
Department of Computing Science
University Of Glasgow
power@dcs.gla.ac.uk

Alice Miller
Department of Computing Science
University Of Glasgow
alice@dcs.gla.ac.uk

**Abstract**

We present a technique for the automated detection of potential structural and data symmetries from a probabilistic specification language. This approach involves the construction of an extended static channel diagram, a graphical representation of channel-based communication and global variable accesses from a specification defined in our custom made language SPS. This graph is used to compute potential symmetries whose validity is verified against the probabilistic specification. Unlike previous approaches this method can detect arbitrary structural and data symmetries.

## 1 Introduction

As software becomes more complex the need for development techniques capable of uncovering errors at design time is critical. A model checker accepts two inputs: a specification $\mathcal{P}$, described in a high level formalism and a set of testable properties, $\phi$. A model checker generates and exhaustively searches a finite state model $\mathcal{M}(\mathcal{P})$ to confirm if a property holds, or conversely, report a violation of the system specification. The intuition is that, bugs found in the model will reveal bugs in the system design. However, the application of model checking is limited as the state- space of even moderately sized concurrent systems can be too large for state-of-the art machines to exhaustively search.

Although verification algorithms have a linear run time complexity, this is offset as the number of states in a model grows exponentially as parameters are added. Consequently, research often focuses on techniques to reduce the impact of the state-space explosion. In the probabilistic domain, research into alleviating the state space explosion problem is still in its infancy. Only two techniques in the form of symbolic state storage [5] and partial order reduction [2] have been comprehensively investigated, and recently steps have been taken in the application of symmetry reduction to symbolic storage schemes [6]. This research is of importance due to the additional overhead required by probabilistic verification algorithms. To this end, we are investigating the application of symmetry reduction in the area of probabilistic explicit state model checking.

## 2 Symmetry Reduction for Probabilistic Model Checking

In model checking, symmetry reduction involves replacing sets of symmetrically equivalent states in a model $\mathcal{M}$ by a single representative, $rep(s)$, from each equivalence class. The resulting structure $\mathcal{M}'$ is called a quotient structure. For highly symmetric systems, exploring the quotient structure only, can result in a reduction factor exponential to the number of system components. In the probabilistic domain a commonly used structure is a Discrete Time Markov Chain (DTMC).

For a DTMC, $\mathcal{D} = (S, s_0, P)$, a permutation $\alpha : S \rightarrow S$ that preserves the transition relation and set of initial states is termed an automorphism of $\mathcal{D}$. The set of all automorphisms of $\mathcal{D}$ forms a group where the operator is a mapping and is denoted $Aut(\mathcal{D})$. For $G \leq Aut(\mathcal{D})$, the orbits of S under G can be used to construct a quotient DTMC $\mathcal{D}_G$. Let $\mathcal{D}$ be a DTMC, and $G$ an automorphism group of $\mathcal{D}$. The quotient structure $D_G = (S_G, s^0_G, P_G)$ is defined as:

- $S_G = \{rep_G(s) : s \in S\}$, where $rep_G(s)$ is a unique representative of $s^G$
- $s^0_G = rep_G(s_0)$
- $P_G(rep_G(s), rep_G(t)) = \sum_{x \in t^G} P(rep_G(s), x)$

For a model $\mathcal{D}$ and its quotient model $\mathcal{D}_G$ with respect to a group G, $\mathcal{D}, s \models \phi \Leftrightarrow \mathcal{D}_G, rep_G(s) \models \phi$ for every symmetric PCTL formula. It follows that $\mathcal{D} \models \phi \Leftrightarrow \mathcal{D}_G \models \phi$ [7]. By choosing a suitable symmetry group $G$, model checking can be performed over $\mathcal{D}_G$ instead of $\mathcal{D}$, often resulting in considerable savings in memory and verication time. If automorphisms can be identified in advance, then a quotient structure can be incrementally constructed even if the original structure is intractable.

# 3 Automated Symmetry Detection

It has been establish [4] that there is a correspondence between symmetries in a specification's underlying communication structure and those in the model. To determine the symmetry present in a system, a structure called a static channel diagram [4], a graphical representation of potential communication within the system, can be generated directly from a model specification. As with most other automated symmetry detection techniques static channel diagrams relate to structural symmetry. However, another form of symmetry, namely data symmetry, can be exploited to increase the effectiveness of model checking.

Specifications often contain large data structures that can be populated by numerous potential values. To capture potential data symmetries in a specification we have extended the definition of a static channel diagram to include nodes for global variables and edges between process identifiers and global variables. An edge between process identifiers and global variable nodes is included if a process can potentially update the variable, and an edge from a global variable node to a process identifier is included if the result of an update may be affected by the value of the variable.

To test our extension a suitable specification language was required. The language would need to be able to define a wide range of probabilistic models, be compatible with existing approaches to automated symmetry detection and have formally defined semantics. To our knowledge the only probabilistic specification language potentially compatible with with existing techniques is ProbMela [1]. However, ProbMela [1] is a language with many constructs, making it infesable to rigourously prove implemented reduction techniques are sound. Therefore, we elected to define our own smaller probabilistic channel based language called, Symmetric Probabilistic Specification (SPS). SPS is tailored to meet all the above criteria and has a fully defined grammar and type system, in addition to providing precise DTMC structure semantics for a specification.

We have created a tool that implements these static channel diagram extensions. An SPS specification is taken as input and its abstract syntax tree constructed. In turn the tree is used to type-check the specification ensuring that variables are used appropriately. If deemed correct the static channel diagram $C(\mathcal{P})$ is generated and saucy [3] used to compute a set of generators for $Aut(C(\mathcal{P}))$. Each of the generators is checked for validity against the specification. These checks are generally conditions on assignments to process id sensitive variables and can be efficiently checked. For an element $\alpha \in Aut(C(\mathcal{P}))$, we say that $\alpha$ is valid (for $\mathcal{P}$) if $\alpha(\mathcal{P}) \equiv \mathcal{P}$ .

The complexity of deriving $C(\mathcal{P})$ from $\mathcal{P}$ is linear in the size of $\mathcal{P}$ but no polynomial time algorithm is known for the calculation of generators for $Aut(C(\mathcal{P}))$. However, saucy [3] was specifically designed to calculate automorphisms of sparse graphs and $C(\mathcal{P})$ tends to be relatively sparse. The performance of the tool is generally very good and a set of valid specification automorphisms can be calculated in under a second.

In current work we are proving the following correspondence theorem. Let $\mathcal{P}$ be a channel based probabilistic specification with extended static channel diagram $C(\mathcal{P})$ and associated DTMC structure $\mathcal{D}$. Let $\alpha \in C(\mathcal{P})$. If $\alpha$ is valid for $\mathcal{P}$ then $\alpha^* \in Aut(\mathcal{D}))$. This theorem will prove the basis for the sound implementation of the first on the fly probabilistic model checker that utilises symmetry reduction to manage the state space.

# References


[1] C. Baier, F. Ciesinski, and M. Großer. ProbMela and verification of Markov decision processes. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):22–27, 2005.

[2] F. Ciesinski and C. Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems.

[3] P. Darga. SAUCY: Graph automorphism tool. *http://www.ec.umich.egpb/auro/s.html*.

[4] A.F. Donaldson, A. Miller, and M. Calder. Finding symmetry in models of concurrent systems by static channel diagram analysis. *Electronic Notes in Theoretical Computer Science*, 128(6):161–177, 2005.

[5] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. *Lecture Notes in Computer Science*, 3920:441, 2006.

[6] M. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. *Lecture Notes in Computer Science*, 4144:234, 2006.

[7] A. Miller and A.F. Donaldson. Property Preservation in Quotient Structures. Technical report, Department of Computing Science, University of Glasgow, 2009.

# Automatic Generation of Dynamic Investigation Problems

Ramin Ramezani and Simon Colton
Computational Creativity Group, Department of Computing, Imperial College, London, UK
raminr,sgc@doc.ic.ac.uk

## 1 Introduction and Motivation

One of the ultimate goals of AI computer programs is to solve real world problems as efficiently, or even better than people; sometimes to even solve problems that cannot be solved by people. Imagine a crime case with many suspects involved, where each of the suspects has various motivations for the murder which makes the case fairly complicated. For instance the amount of information may be too large for a detective to process. Considering that the knowledge about the crime may not even be sufficient for the detective to deduce the murderer, he/she may refer to previously solved cases which bear resemblance to the current one, hoping to find information that can be generalized to the present problem. Employing this new information may lead to identifying the murderer or to at least making it easier by excluding some of the suspects. We call such problems *investigation problems*, (IPs). These may exhibit ambiguity and complexity but AI problem solving techniques such as machine learning, constraint solving and automated theorem proving are considered as powerful tools for solving such problems. In this paper we present a formalization of IPs and a way of generating them. Furthermore, we will discuss an experiment in which different scenarios of a certain IP is generated and is solved by a Constraint Satisfaction Problem (CSP) solving approach.

## 2 Definition of Dynamic Investigation Problems (DIP)

Before defining IPs, it is essential to know the 1-connectedness definition: (Colton and Muggleton, 2006)
Suppose $C$ is a clause of the form: $P_i(X_1, \ldots, X_m) :- P_1(Y_{11}, \ldots, Y_{1n_1}), \ldots, P_l(Y_{l1}, \ldots, Y_{ln_l})$ where each $X_i$ is a variable and each $Y_{ij}$ may be a variable or a ground term. Then the variable $V$ which is a literal in a body of $C$ is said to be $1\_connected$ if it satisfies below conditions:

- $V = X_1$ or
- $\exists\, i, j, k \quad s.t \quad j \neq k$, $Y_{ij} = V$ and $Y_{ik} = X_1$ or
- $\exists\, i, j, k \quad s.t \quad j \neq k$, $Y_{ij} = V$ and $Y_{ik} = X_1$ is a 1- connected variable

We consider investigation problems as being similar to CSPs with a finite set of variables, each associated with a finite domain and a set of constraints; the difference is that the background knowledge of an IP contains information about past cases, in addition to the current case description. Such close shady information may contain the same set of constraints and variables as in the current case, partially or completely. Past cases contain hypotheses that can be interpreted as embedded constraints and these constraints can become explicit using machine learning techniques.

Let $X \in \{x_1, x_2, \cdots, x_n\} = D_x$, where $D_x$ is the domain of $X$ and $y_1, \cdots, y_n$ are domain values. In a general CSP problem with one variable, the task is to assign a value from domain $D_x$ to $X$ such that all the constraints (see below) are satisfied simultaneously. However, for an IP, the problem is to identify an ordered list of domain values for $X$. Let $[g'_1, g'_2, \cdots, g'_k] \subseteq D_x$, where each $g'_i \in D_x$ and carries a likelihood degree. The likelihood degree of $g'_i = \frac{\text{Number of constraints satisfied by } g'_i}{\text{Total number of constraints}}$. The constraints being satisfied by $g'_i$ are either from problem constraints or from the past cases' embedded constraints. The more constraints each domain value satisfies, the higher the likelihood it has for being the correct answer, thus the answers are prioritized.

Let $C = \{C_1, C_2, \cdots, C_m\}$ be a set of constraints declared in the problem definition such that each constraint $C_i$ contains predicates and variables. Define $Pred\_C_i$ to be a set of all predicate names appearing in the constraint $C_i$ and every $C_i$ is in the form:
$C_i(X) :- P_{i1}(Y_{11}, \ldots, Y_{1n}), \ldots, P_{ia}(Y_{a1}, \ldots, Y_{al})$ where:

- $a, l, n$ are arbitrary finite values
- The arguments in $P_{ij}$ may each be a variable or a ground term.
- $\forall \quad P_{ij} \quad s.t \quad 1 \leq j \leq a$, $P_{ij}$ should have a variable $V$ which is 1-connected.

Let $Const\_C_i$ be the set of all constants appearing in $C_i$. Let $E = \{E_1, E_2, \cdots, E_p\}$ be a set of past cases. We define every past case $E_i$ as a set of $\{p_1, p_2, \cdots, p_n\}$ where each $p_i$ is a ground predicate. There are embedded constraints in

every $E_i$ which will become explicit. Let $Pred\_E_i$ be the set of all predicate names appearing in the past case $E_i$. Let $Const\_E_i$ be the set of all constants appearing in the past case $E_i$. Let $Pred\_C_T$ be the union of all the predicates present in each constraint, $Const\_C_T$ be the union of all the constants present in each constraint and $Pred\_E_T$ be the union of all the predicates present in each of the past cases.

### 2.1 Conditions for being an IP

(i) $\forall j\,(\exists\, S \subseteq Pred\_E_j \quad s.t \quad S \subset Pred\_C_T)$

There should be an overlap between the predicates present in every past case and the overall predicates appearing in the constraints. The commonality confirms the relevance between a past case and constraints.

(ii) $\forall j\,\exists\, S' \subseteq Const\_E_j \quad \wedge \quad (\exists\, M : S' \rightarrow (D_x \cup Const\_C_T)) \quad s.t$
$\Big((\exists \delta \in S',\, \exists \lambda \in (D_x \cup Const\_C_T)) \quad s.t \quad M(\delta) = (\lambda)\Big)$

Note that $M$ is a mapping function. This condition implies that at least one constant in a past case can be mapped to a value in the problem domain.

A *dynamic* investigation problem is similar to an IP. The only difference is an additional time aspect in the problem. The background knowledge keeps changing over time due to addition or extraction of constraints and past cases at different instances. With change in time, constraints, past cases and the domain of the variable can be modified. Let $T$ be a finite set $s.t \;\; T = \{t_1, t_2, \cdots, t_n\}$ where $n \geq 2$ represent time instances. At each time instances the conditions of an IP should be satisfied. Over time, constraints and past cases can be completely altered, however, assuming $D_{1x}$ to be the domain of $X$ at time $T = t1$ and $D_{2x}$ to be the domain of $X$ at time $T = t2$, the following condition should be always satisfied: $\exists\, y \in D_{1x}$ such that $y \in D_{2x}$.

## 3 DIP Constraints Generation

To show that an investigation problem can be defined in a DIP form and is ultimately amenable to a constraint solving approach, we considered a board game known as Cluedo. In this game, the player moves around a mansion with nine rooms where the murder can take place and collects clues to infer which suspect has murdered the victim. In the classic single player cluedo game, the player tries to determine the identity of the murder by searching every single room. The information gathered by the player at each step provides the constraints and the information about a DIP at time $T$. We wrote a program using Prolog to randomly generate different scenarios for a Cluedo game. The program outputs a limited set of constraints and predicates at each time slot, making it similar to the real game in which the player can only collect few clues at every room. However, it is worth pointing out that we only focused on the current case of a DIP, therefore, random generation of the previous cases has not yet been taken into consideration.

To make the game more interesting, the amount of information being generated at each step is also varied. Hence the player may find more evidence in a room and less in another. In addition, we increased the complexity of the problem by adding constraints and predicates about the suspects and the murder case in general. For instance *"the murderer should be tall and quick"* or *"the murderer should be angry with something"* are added as constraints. The added predicates could be complementary to the constraints, like *"Professor Plum is tall"* or *"Scarlett is short"*. Below is the partial output of the DIP Cluedo generator:

$at\,Time\,T = 4: \quad is\_angry(scarlett). \quad was\_found(revolver, patio). \quad murderer(X) : -is\_tall(X).$

To show that the DIP is amenable to a constraint solving approach, we wrote a CSP in the syntax of Sicstus CLPFD (Carlsson et al., 1997). At each step, we fed the output of the DIP to the CSP and showed that the solver can not come up with a single solution only until the last stage. This means that all the constraints and background information are needed for the program to infer the murderer, murder scene and murder weapon. The output of the final stage depending on the scenario being generated might be: *"Scarlett committed the murder in the Kitchen using a Candlestick"*. However, the program at each stage, depending on the available information, can assign a likelihood degree to each suspect. In addition, if the generated data is not sufficient to solve the mystery even at the final stage, using the previously solved cases could help us to learn a set of rules that can replace the missing information (Ramezani and Colton, 2009) and we intend to pursue this in the future.

## Acknowledgments

This work is funded by EPSRC grant EP/F036647.

## References

M. Carlsson, G. Ottosson, and Bjrn Carlson. An open-ended finite domain constraint solver. 1997.

S. Colton and S. Muggleton. Mathematical applications of inductive logic programming. *Machine Learning*, 64(1-3): 25–64, 2006.

R. Ramezani and S. Colton. Solving mutilated problems. *Automated Reasoning Workshop*, 2009.

# Reasoning with equality in a contextualised inference system

Allan Ramsay
*School of Computer Science, University of Manchester, Manchester M13 9PL
Allan.Ramsay@manchester.ac.uk

**Abstract**

The aim of the work reported here is to support the kind of epistemic planning and plan recognition required for generating and interpreting goal-driven linguistic actions. Planning with linguistic actions poses a number of problems that do not arise in most other areas where planning is required. These are compounded when the conversation involves reasoning about whether two terms refer to the same entity. The current paper addresses the problem of reasoning about equality across contexts.

## 1 Planning with linguistic actions

The work described here is aimed at the task of choosing appropriate linguistic actions for influencing your hearer's actions, and of recognising intentions behind the speaker's linguistic actions. This is a well-known task (Allen and Perrault, 1980; Cohen and Levesque, 1980; Appelt, 1985), but there are a number of pitfalls that are under-estimated in the early work in this area. The key problem is that the range of easily distinguishable action schemas is very limited (in English, for instance, there are four recognisably different action types–statements, commands, polar questions and WH-questions), each of which can be instantiated in a very wide range of ways. The effect of a given instantiation of one of the schemas depends almost entirely on how it is instantiated (so the effect of stating that there is a pile of wet washing to be hung up will be very different from the effect of stating that I have hung up the washing) and on the beliefs of the participants in the dialogue (so telling you that there is a pile of washing to be hung up in a situation where you have asked me what jobs there are will be very different from doing the same thing in a situation where you already know that this is true) (see (Bunt, 2000) for further discussion of this problem).

The effects of linguistic actions are thus very unpredictable, so that choosing an action to achieve a given goal is much more difficult than is the case for actions in most other domains. In particular, static analysis of the problem space (Kambhampati, 1997; Nguyen and Kambhampati, 2001) is impossible, since you cannot tell whether the effects of one action will entail the preconditions of another unless you know a great deal about the context. The obvious way to get you to believe some proposition $P$, for instance, is to tell you that it is true. Unfortunately, this will only work if you regard me as a trustworthy source of information in the domain of $P$. It may therefore be more effective to tell you about some other proposition $P'$, where I believe that your beliefs would enable you to derive $P$ from $P'$, and where I believe that you would view me as a reliable source of information about $P'$.

We have described elsewhere a combined inference engine and planner where actions are chosen by the planner by checking whether their direct effects *entail* the desired goals (Field and Ramsay, 2004b; Ramsay and Field, 2006b), and we have shown how this can be used to reason about complex linguistic activities such as lying, telling jokes and being sarcastic (Field and Ramsay, 2004a; Ramsay and Field, 2006a, 2008). The essence of this planner is that we allow hypothetical reasoning, where the gaps in a proof are collected and used to instantiate the linguistic actions. Suppose, for instance, that I wanted you to believe $P$, and I believed that you believed $P' \rightarrow P$. Since my goal is for you to believe $P$, my first step would be to see whether I thought you already believed it (since if that were true, there would be no need for me to do anything). So I would try to prove $bel(you, P)$. This proof *would have* succeeded if I had found that $bel(you, P')$ was true. So this would be recorded as a hypothesis, and would be a candidate for something that I might tell you. If I felt that you would be more likely to trust me as a source for $P'$ than for $P$ itself, telling you $P'$ would be a better idea than telling you $P$.

## 2 Contexts as labels

There have been numerous attempts to mix epistemic reasoning and reasoning about change (Moore, 1984; Chapman, 1987). The majority of these follow Hintikka (1962) in treating epistemic logic as a form of modal logic, and then assuming a possible worlds semantics for modal logic. We believe that this is fundamentally the wrong approach, since it inevitably and inextricably leads to logical blindness (if $P$ and $P'$ are equivalent and $X$ currently believes $P$ then $X$ must also *currently* believe $P'$) and logical omniscience (if $X$ believes $P' \rightarrow P$ and $X$ currently believes $P'$ then $X$ must also

*currently* believe $P$). Of course, no actual implementation of an inference engine for modal logic has these consequences, because all such inference engines are resource-bounded. Nonetheless, to take a fundamentally incorrect theory of belief and then to rely on the fact that your implementation of that theory is incomplete to get you to a more acceptable situation does not seem to be the best way to go.

We therefore take an intrinsically proof theoretic view of belief. Belief is something you *do*, not something you *have*. In that case, if I want to reason about whether you believe something I should think about what I would do if I were you. If I have a set of propositions and axioms that I believe are available to you, then I should see what I would be able to do with those propositions and axioms if I were you (which might involve using a different set of inference rules from the ones I would use myself, if I thought for instance that you were particularly gullible). This is the approach taken by Konolige (1986), which we believe is more appropriate for reasoning about belief than treating it as a modal logic and explicitly reasoning about sets of possible worlds. We implement this notion by including the context in which a proposition or axiom is available as an element of its label (Gabbay, 1996). The details of the inference engine, and its extension to the untyped intensional logic 'property theory' (Turner, 1987), are given in (Ramsay, 2001), and the linkage between the inference engine and the planner are given in (Ramsay and Field, 2006b).

# 3 Equality in alternate contexts

We need to allow for reasoning about equality in this framework. The information that is conveyed in dialogue oftens concerns the fact that two descriptors denote the same individual:

A: Who is the president of the ICCL?
B: Martin Kay.

Clearly, at the start of this dialogue A's model of the word contains a representation of the president of the ICCL. Equally clearly, B believes that A's model of the world contains a representation of an individual called Martin Kay, or he would not think that it was appropriate to use the name *'Martin Kay'* as a descriptor. The function of B's utterance is to inform A that these two entities are the same–to tell him that they are equal. After this, when A thinks about the president of the ICCL he can deploy all the facts that he has about Martin Kay (e.g. he will be able to answer the question *'Is the president of the ICCL a man or a woman?'*).

The kind of reasoning required here is not complex. Most of the equalities we are concerned with arise as a consequence of simple statements of identity, rather than as a result of reasoning with sets of rewrite rules. Thus in some ways our task is easier than the task tackled by theorem provers whose primary goal is to obtain identities on the basis of the rules of some branch of mathematics. At the same time, we have to cope with the fact that different identity statements are available in different contexts (e.g. to different people, as in the dialogue above). We therefore include chains of equivalences, annotated with the contexts in which they are available, in the label associated with a branch of a proof. When we want to unify two terms, we invoke the equivalences that are available in the current context to rewrite each term dynamically to the canonical form in the current context. This allows us to deal with situations where one person believes that $X$ and $X'$ are different entities and another believes that they are the same, simply by interfering with unification algorithm.

# References


J F Allen and C R Perrault. Analysing intention in utterances. *Artificial Intelligence*, 15:148–178, 1980.

D Appelt. Planning English referring expressions. *Artificial Intelligence*, 26:1–33, 1985.

H C Bunt. Dialogue pragmatics and context specification. In H C Bunt and W J Black, editors, *Abduction, Beliefs and Context: Studies in Computational Pragmatics*, pages 81–151, Amsterdam/Philadelphia, 2000. John Benjamins.

D Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

P R Cohen and H Levesque. Speech acts and the recognition of shared plans. In *Proceedings, Canadian Society for Computational Studies of Intelligence*, pages 263–270, 1980.

D G Field and A M Ramsay. Sarcasm, deception, and stating the obvious: Planning dialogue without speech acts. *Artificial Intelligence Review*, 22:149–171, 2004a.

D G Field and A M Ramsay. How to build towers of arbitrary heights. In *The 23rd Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2004)*, University College Cork, 2004b.

D M Gabbay. *Labelled Deductive Systems*. Oxford University Press, Oxford, 1996.

J Hintikka. *Knowledge and Belief: an Introduction to the Two Notions.* Cornell University Press, New York, 1962.

S Kambhampati. Refinement planning as a unifiying framework for plan synthesis. *AI Magazine*, 18(2):67–97, 1997.

K Konolige. *A Deduction Model of Belief*. Pitman, London, 1986.

R C Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358, New Jersey, 1984. Ablex Pub. Corp.

X Nguyen and S Kambhampati. Reviving partial order planning. In *IJCAI*, pages 459–466, 2001.

A M Ramsay. Theorem proving for untyped constructive $\lambda$-calculus: implementation and application. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(1): 89–106, 2001. ISSN ISSN: 1367-0751.

A M Ramsay and D G Field. How to change a person's mind: understanding the difference between the effects and consequences of speech acts. In *5th International Conference on Inference in Computational Semantics (ICoS 06)*, pages 27–36, Buxton, 2006a.

A M Ramsay and D G Field. Planning ramifications: when ramifications are the norm, not the problem. In *11th International Workshop on Non-monotonic Reasoning (NMR-06)*, pages 344—351, Ambleside, 2006b. AAAI.

A M Ramsay and D G Field. Speech acts, epistemic planning and Grice's maxims. *Logic and Computation*, 18:431–457, 2008. doi: 10.1093/logcom/exm073.

R Turner. A theory of properties. *Journal of Symbolic Logic*, 52(2):455–472, 1987.

# Towards Balanced Distribution of Computations through Automated Reasoning

Reza Shojanoori
University of Westminster
Shojanr@my.wmin.ac.uk

Radmila Juric
University of Westminster
Juricr@wmin.ac.uk

Mahi Lohi
University of Westminster
lohim@wmin.ac.uk

**Abstract**

The penetration of a wide variety of networking devices, wireless networks, personal digital assistants, sensors, actuators, and numerous other mobile embedded devices into the field of computing, together with their increased deployment by users, have all led to 'ubiquitous computing' (Weiser, 1993) as a reality. We have moved towards an 'unnoticeable' computing environment, which serves people in their everyday life, necessitates 'smart' or 'intelligent' spaces.

## 1 Introduction

The type of computation required in such spaces (Intille, 2006) is diverse and semantically rich. We have developed a software application which enriches such spaces in terms of automating reasoning mechanisms for facilitating decision making. To create such a semantically rich computing environment we have accommodated the power of semantic web standards, tools and technologies. Situations in such spaces change constantly hence any dynamic decision making depends on a number of variables, value of which have to be acquired. We have to pay attention on how these variables are defined. They may be basic facts, "sensed", given, or inferred. They may be a consequence of running a "rule", which in turn may be based on a number of other axioms. They are all being used and manipulated in the architecture to determine the reusability, ubiquity, and scalability of the solution.

## 2 Reasoning in Software Application

We have used the architecture shown in Figure 1 in different ways to get closer to our desire towards a consensus on what computation is in semantic environments, how much we depend on automated reasoning and how crucial is to make correct decisions on what exactly data and computations are in pervasive spaces.

The core of this architecture is based on Ontology of the domain of interest, and an Inference Mechanism. This core is utilised in a pervasive computing environment by applications designed for the smart environment to dynamically respond to the context changes in various situations. The Inference Mechanism is responsible for reasoning upon the 'context' using predefined set of rules. The Context layer is responsible for collecting and interpreting data derived from sensors. The Ontology needs this information to make provision for the reasoning mechanism upon a particular situation.

We have designed the ontology, defined the rules and developed the application in three different ways to define the consensus of what computation is in semantic spaces.

1. we use the ontology as a hierarchy of classes with minimal use of ontological restrictions. This has led to extensive use of reasoning rules outside the ontology and therefore the application is heavily overloaded. In this approach non-ontological rules are used to reason automatically at different levels.
2. we "normalize", i.e. 'normalised' ontology (Rector, 2002) has been developed with as much necessary restrictions as possible. The purpose was to minimise dependency of the solution given by reasoning rules and reduce the level of automated reasoning at the application level.
3. we decide to exploit 1 and 2 above and create "the third way", which is somewhere between the first two. We advocate a moderate use of reasoning rules, i.e. they are defined and run only when they have to complement the ontology. Therefore automated reasoning at the application level, which is generally easy to implement, exists only when consequent running of the ontology, for classification of its concepts, in one particular "situation", is necessary.

We try to bring more balanced computational power between the Core and the Application itself. We would like to see how much computation (reasoning!) could be removed from the Application and placed within the Core in order to exploit the power of semantic technologies. We do not see application as the only means of building intelligent spaces and performing automated reasoning through semantic technology. The purpose of our work is to illustrate the amount and the type of computational code based on automated reasoning which is relevant for managing ontological concepts and consequently delivering appropriate services to inhabitant of smart spaces (Shojanoori et al, 2010).

We have used Protégé-OWL to develop ontology, Pellet reasoner to classify ontology and infer new knowledge, and SWRL language for rules to complement OWL-DL where it falls short of expressing the environment. The application is written in Java and communicates with the ontology using OWL-API.

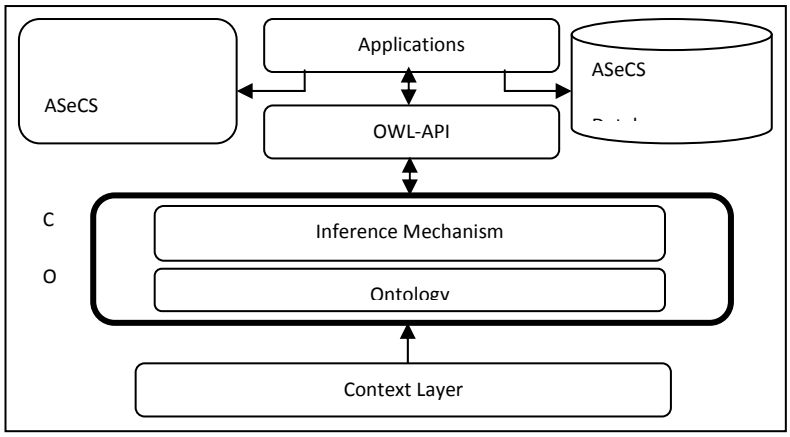



Figure 1: Smart Space Example Architecture

# References

Stephen Intille. The Goal: Smart People, Not Smart Homes. *Proceedings of the 4th International Conference on Smart Homes and Health Telematics*, 26-28 June, 2006, Belfast, Northern Ireland, UK, pp.3-6.

Alan Rector, Normalisation of Ontology implementations: Towards modularity, re-use, and maintainability. In EKAW Workshop on Ontologies for Multiagent Systems, 2002.

Reza Shojanoori, Radmila Juric, Babak Tourani. Experience of Building Assisted Self Care Systems Within Smart Home Environment. To appear in Proceedings of the 15th International Conference on System Design and Process Science, June 2010, Dallas, US.

Mark Weiser. Hot topics-ubiquitous computing, *Computer*, Vol.. 26, Issue 10. 1993, pp. 71-72

# A Flexible Framework for Experimental Mathematics

Osama Taleb* and Volker Sorge*
*School of Computer Science
University of Birmingham
{O.Taleb|V.Sorge}@cs.bham.ac.uk

## 1 Introduction

Over the last decade several environments and formalisms for the combination and integration of mathematical software systems have been proposed. Many of these systems aim at a traditional automated theorem proving approach, in which a given conjecture is to be proved or refuted by the cooperation of different reasoning engines. However, they offer little support for experimental mathematics in which new conjectures are constructed by an interleaved process of model computation, model inspection, property conjecture and verification using state-of-the-art symbolic reasoning system. For example, Bertoli et al. (1999) presented the OMRS/OMSCS framework which provides a theoretical approach to combine theorem proving and symbolic computation tools, that has the drawback that a system integration according to this framework requires the re-implementation of the systems to be combined. On the other hand, Zimmer and Kohlhase (2002) MathWeb Software Bus allows more readily to integrate existing systems but is still primarily geared towards supporting single proof problems only. More recently, Charnley (2010) developed the GC toolkit which provides a framework for developing combined reasoning systems based on the cognitive theory of Global Workspaces. The tool allows users to integrate systems with their own bespoke reasoning algorithms using a graphical user interface. But since GC's primary aim is the distribution of reasoning via competing processes it offers limited control of how systems collaborate and requires a relatively high user knowledge of the integrated reasoning systems. Both are essential drawbacks for experimental mathematics, which often requires tight process control when specifying an experiment as well should allow the user to focus on the mathematical problem at hand rather then the underlying logic of the systems involved.

The goal of our work is therefore to develop and implement a system that allows a user to easily experiment with the combination of symbolic reasoning systems in different application scenarios. This is best achieved in a visual programming environment that enables the combination and re-combination of systems in a plug and play fashion. Once the experiments have yielded a desirable combination of reasoning components it should be possible (1) to encapsulate and characterise the combination as a new component within the environment for duplication and reuse and (2) to generate an efficient stand-alone system from the specification. We currently develop a system as an extension of Simulink – Karris (2006) – within the Matlab environment. Simulink is an interactive and customisable graphical environment which serves us as a robust basis providing all the basic features we are looking for. This frees us to concentrate on the scientifical problems of how plug-and-play interaction between symbolic reasoning systems can be achieved without requiring the user to have detailed knowledge of the underlying processes.

## 2 Combining Systems for Experimental Mathematics

The following is a list of symbolic reasoning systems we want to integrate together with a brief description of some of their non-standard usage we envision.

**Model Generators** find examples or counter examples for first order formulas, which can be further processed and passed to other reasoning systems. They can also purely be used as filter, depending on whether or not some model exists.

**Finite Domain Solvers** SAT, SMT, and Constraint Solvers can be used to both find examples and prove conjectures in finite domains as well as perform filter functionality.

**Automated Theorem Provers** can be exploited to provide proofs as artefacts in their own right and to filter first order conjectures wrt. their validity.

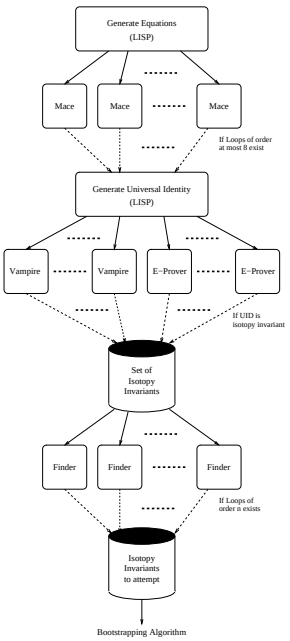
**Symbolic Computation Engines** Computer Algebra or bespoke user algorithms can be employed to manipulate and solve equations, rewrite expressions, or reduce the complexity of the problem to make them amenable by other systems.

**Machine Learning Systems** can be used for concept or automated theory formation and to mine and learn from data accumulated during experiments.

**Data Stores** act as data sources and sinks to collect and collate results.

# 3 Case Studies

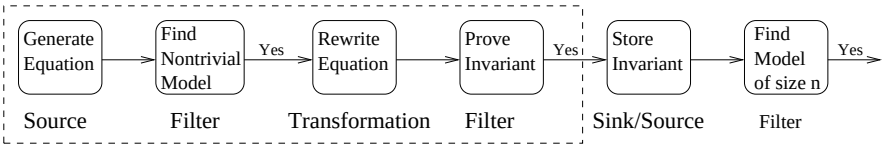
We will develop our framework by drawing on the experience of several case studies for the integration of heterogeneous reasoning systems. Consequently, one measure for our success will be that one can easily re-build those systems in our framework.
For example, Sorge et al. (2006) describes the implementation of a system which combines mathematical object generation, transformation and filtering, conjecture generation, proving and disproving for mathematical discovery in non-associative algebra. While the system has generated novel, fully verified theorems, their construction involved a lot of ad hoc communication between disparate systems. Furthermore the system consists of a number of sub-systems for bespoke tasks that are not only combinations of various systems, but are also interesting in their own right.
The diagram on the right depicts one of these sub-systems which is used to generate one particular type of algebraic invariant that can be used in the overall discovery process. Without presenting any of the mathematical details the idea of the setup is as follows: (a) We systematically generate universally quantified equations, and (b) check whether some have valid models for our algebraic domain using model generation. (c) Those equations that indeed have a model are rewritten to an extended form, (d) and checked whether or not they can be used as invariant by employing first order theorem provers. (e) Valid invariants are collected and (f) filtered again depending via model generation to determine whether they can be employed during one particular discovery step.




# 4 A Graphical Environment for Specifying Combinations

We will develop our system as an extension of Simulink within the Matlab environment. Simulink provides an interactive graphical environment and a customisable and extensible set of block libraries. It has full access to Matlab's features, including its programming language and its ability to integrate code in other languages such as C and Java. In particular we can embed various systems via Matlab S-functions and use Matlab's object oriented facilties to implement the datatypes required by the symbolic reasoning systems as classes. We can model state-based behaviour of the embedded systems using the Stateflow Toolbox and we can exploit symbolic computation facilities in the computer algebra system MuPad via the Symbolic Math Toolbox. Furthermore, Simulink can not only execute the symbolic reasoning process in the environment itself but also automatically generate runnable C code from the block diagrams to provide efficient stand-alone systems. Thus we have a robust and well-maintained bases on which to build our system that provides all the basic features we are looking for and thus frees us to concentrate on the scientifical problems that need to be solved.

In this environment the case study example can be modelled with the following data flow diagram below. While this schematic depiction omits the necessary background formalisations, we currently distinguish three language levels in the system: (1) The inter-system communication language, (2) the language for describing computations, objects of communications as well as the theory environment a particular block is embedded in, (3) and the specification language for mathematical formulations via the user interface. For their formalisation we aim to exploit biform theories similar to Carette et al. (2007).




# References


P. Bertoli, J. Calmet, F. Giunchiglia, and K. Homann. Specification and Integration of Theorem Provers and Computer Algebra Systems. *Fundamenta Informaticae*, 39:39–57, 1999.

J. Carette, W.M. Farmer, and V. Sorge. A rational reconstruction of a system for experimental mathematics. In *Proc. of Calculemus 2007*, volume 4573 of *LNCS*, pages 13–26. Springer Verlag, 2007.

J.W. Charnley. *A Global Workspace Framework for Combined Reasoning*. PhD thesis, 2010.

S.T. Karris. *Introduction to Simulink with Engineering Applications*. Orchard Publications, 2006.

V. Sorge, A. Meier, R. McCasland, and S. Colton. Automatic construction and verification of isotopy invariants. In *Proc. of IJCAR 2006*, volume 4130 of *LNAI*, pages 36–51. Springer Verlag, 2006.

J. Zimmer and M. Kohlhase. System Description: The MathWeb Software Bus for Distributed Mathematical Reasoning. In *Proc. of CADE-18*, volume 2392 of *LNAI*. Springer Verlag, 2002.

# Heterogeneous Reasoning in Real Arithmetic

Matej Urbas*
*Computer Laboratory
University of Cambridge
Matej.Urbas@cl.cam.ac.uk

Mateja Jamnik†
†Computer Laboratory
University of Cambridge
Mateja.Jamnik@cl.cam.ac.uk

**Abstract**

Diagrams often complement sentential proofs in mathematics. However, diagrams are rarely used as standalone reasoning tools. Thus we propose to integrate diagrammatic reasoning with an existing sentential theorem prover, thus enabling so-called heterogeneous reasoning, particularly in real arithmetic. We will study a set of diagrammatic proof examples from which we will construct a diagrammatic language, inference rules and communication procedures between the diagrammatic and sentential reasoners. The resulting framework will allow the use of diagrammatic proof steps in the same way as the sentential ones, all within the same attempt to construct a proof.

## 1 Introduction

Most diagrammatic reasoning approaches are strictly informal (e.g., sketches or specific illustrations of a general problem). This lead to numerous diagrammatic formalisation efforts (Hammer, 1994; Howse and Stapleton, 2008; Jamnik et al., 1999). However, proofs "on paper" rarely consist exclusively of drawings. Diagrams are often accompanied by sentential formulae. This motivated some to investigate heterogeneous reasoning (Barker-Plummer and Etchemendy, 2007).

Our goal is to introduce diagrammatic reasoning techniques into an *existing* sentential theorem prover, thus devising a heterogeneous reasoner. We first study heterogeneous proof examples[1] in real arithmetic, from which we will then construct diagrammatic inference rules and language. Finally we will integrate the two modes of reasoning – the diagrammatic logic into the sentential prover.

One of our goals is to show whether heterogeneous reasoning can improve proof intuitiveness in sentential provers. We also believe that heterogeneous methods can provide better or entirely novel proof hints. Hints in homogeneous sentential (Ireland et al., 1999) systems are provided in residual statements of an unsuccessful proof attempt. The unresolved statements can be inspected for clues on how to proceed. However, such hints are often not easily discernible even for experts.

Additionally, naive general inferences from specific diagrams can result in incorrect conclusions.[2] It is thus essential to provide a suitable diagrammatic formalism. Our aims can be broken down into several sub-goals:

**Heterogeneous formalisation.** Introduce diagrammatic inference rules for our logic, check soundness and ensure that the language is powerful enough to cover a sufficiently large subset of problems in the target domain. This will provide a formal connection between the diagrammatic and sentential logics.

**Diagrammatic reasoner.** Construct a diagrammatic reasoner that can either prove a goal or produce a transformed one, which can again be used in the sentential reasoner or act as a hint.

**Integration.** We have to establish a bidirectional translation between the two modes of reasoning and integrate the diagrammatic reasoner into the chosen sentential theorem prover. The truth of all statements must be preserved during the translation. Also, integration must allow not only diagrammatic proof steps, but also conversion of theorems and statements between the two realms.

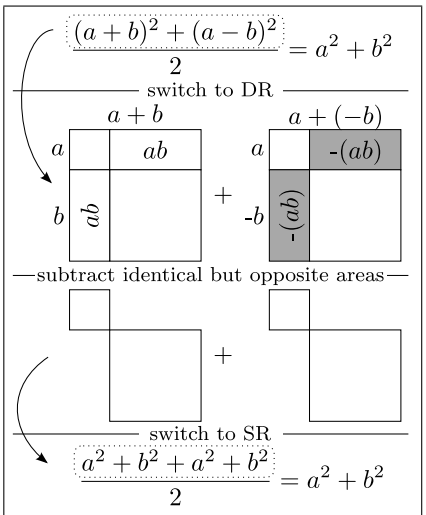



**Figure 1.** Diagrammatic statement rewrite. Gray denotes the opposite sign of the area – same size areas of opposite sign can cancel each-other

## 2 Heterogeneous Reasoning

We examine heterogeneous proof examples to identify the types of diagrammatic/sentential interactions:

[1] Examples were taken from Nelsen's *Proofs without words* (Nelsen, 1997).

[2] An example is Cauchy's erroneous proof of the Euler characteristic for all polyhedra (Lakatos, 1976). This "proof" was unchallenged for decades.

**1.) Statement transform.** Diagrammatic transformations are used to rewrite a sentential statement into some other equivalent statement (Fig. 1).

**2.) Introduction of new goals.** Diagrammatic transformations introduce a set of new sentential statements or lemmas (Fig. 2).

**3.) Proving lemmas.** Entire lemmas can be proved with diagrammatic methods. Formula $\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{2^i} = 1$ is an example of this type. Fig. 3 illustrates a diagrammatically proved lemma for this theorem. Afterwards we use the sentential reasoner to prove $\lim_{n\to\infty} \frac{1}{2^n} = 0$. This last sentential rule is then used to eliminate $\mathrm{B} = \frac{1}{2^n}$ in the diagrammatic proof.

Our target domain is the field of real arithmetic, that is formulae from the ordered field $[\mathbb{R}, +, \cdot, 0, 1, <]$. Numbers and variables are represented as edges and rectangular areas. Areas also act as multiplication of edges. Summation is represented by multiple areas and connected edges that extend in the same direction. Also, we use gray to denote the sign of objects. Universal quantification is implicit in the diagram for all variables.

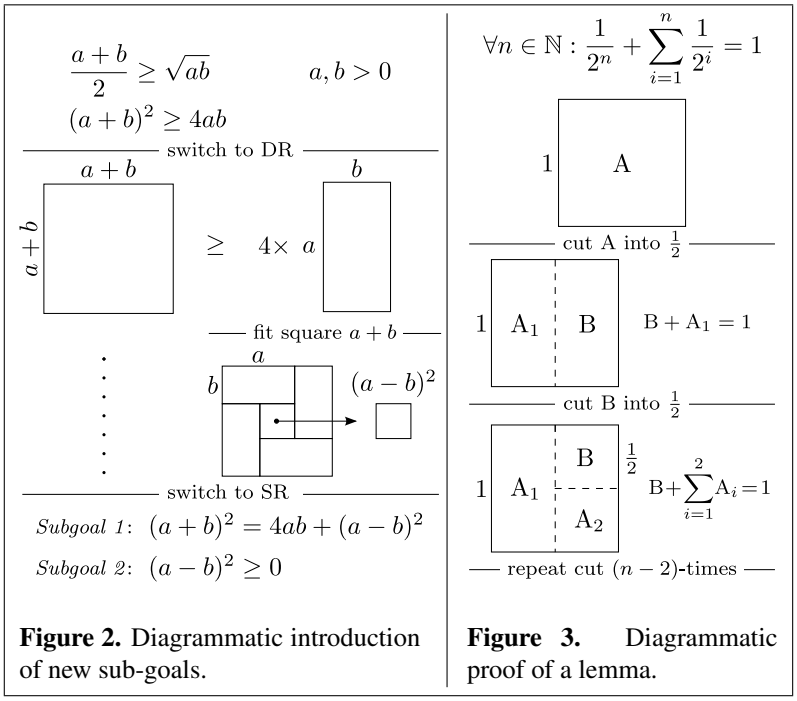



**Figure 2.** Diagrammatic introduction of new sub-goals.

**Figure 3.** Diagrammatic proof of a lemma.

# 3 Methodology

In order to devise a framework with which we can construct heterogeneous proofs as the ones described above, we need to complete the following tasks:

• Define a precise description of the diagrammatic language and its formal inference rules. We will study several examples to determine the required features of the language and the set of inference rules.

• The next step will entail a study of the reasoning and theory formalisms in the sentential reasoner. With this, we will determine how the logic of the reasoner influences our diagrammatic language. We chose Isabelle (Wenzel et al., 2008) as the underlying sentential theorem prover.

• In the last phase, we will design a link between the diagrammatic and symbolic representations. We have chosen a heterogeneous framework architecture where the diagrammatic tactics and statements represent extensions to the built-in native symbolic set of instructions in Isabelle. This will require translation or reuse of internal structures of Isabelle.

In summary, there are many ways in which heterogeneous reasoning can complement sentential approaches, e.g.: more intuitive proofs and proof hints, novel proof tactics, and greater expressive power. We believe that extending a sentential theorem prover with diagrammatic reasoning is viable and advantageous.

# References


Dave Barker-Plummer and John Etchemendy. A Computational Architecture for Heterogeneous Reasoning. *JETAI*, 19(3): 195–225, 2007.

Eric Hammer. Reasoning with Sentences and Diagrams. *NDJFL*, 35(1):73–87, 1994.

John Howse and Gem Stapleton. Visual Mathematics: Diagrammatic Formalization and Proof. *LNCS*, 5144:478–493, 2008.

Andrew Ireland, Michael Jackson, and Gordon Reid. Interactive Proof Critics. *JFAC*, 11(3):302–325, 1999.

Mateja Jamnik, Alan Bundy, and Ian Green. On Automating Diagrammatic Proofs of Arithmetic Arguments. *JOLLI*, 8 (3):297–321, 1999.

Imre Lakatos. Proofs and Refutations: The Logic of Mathematical Discovery, 1976. ISBN 0521290384.

Roger B. Nelsen. Proofs without Words: Exercises in Visual Thinking, 1997.

Makarius Wenzel, Lawrence C. Paulson, and Tobias Nipkow. The Isabelle Framework. In *TPHOLs*, pages 33–38, 2008.