ARW 2012

Proceedings of the 19th Automated Reasoning Workshop

2–4 April 2012

School of Computer Science
The University of Manchester
United Kingdom

Renate A. Schmidt, Fabio Papacchini (Eds.)

Preface

This report contains the proceedings of *The 19th Automated Reasoning Workshop (ARW 2012)*. The workshop was held at the University of Manchester from 2 to 4 April 2012. ARW provides an informal forum for the automated reasoning community to discuss recent work, new ideas and applications, and current trends. It aims to bring together researchers from all areas of automated reasoning in order to foster links and facilitate cross-fertilisation of ideas among researchers from various disciplines, from theoreticians, from implementers and from users of automated reasoning methodologies. Since 1994, when it was held in Leeds as part of the 1994 AISB Workshop and Tutorial Series, ARW has been organised on an annual basis.

We received a total of 25 submissions, some from as far away as Brazil, Finland, France, and The Netherlands. All submissions were accepted for presentation as short talks and posters after a light-weight reviewing round. The programme also included three invited talks: *The Antikythera Mechanism and the Early History of Mechanical Computing* by Mike Edmunds (University of Cardiff, BCTCS invited speaker), *Formal Verification of Software Product Families* by Reiner Hähnle (Technische Universität Darmstadt, BCTCS/ARW joint invited speaker), and *SAT over an Abstract Domain* by Daniel Kroening (University of Oxford, ARW invited speaker). This year the workshop was collocated with *The 28th British Colloquium for Theoretical Computer Science* allowing participants of ARW and BCTCS to attend parallel sessions of the other event.

There were many people who helped organising ARW 2012. First, I would like to thank the authors and presenters of the abstracts, short talks and posters, as well as the invited speakers for their contributions. I thank the ARW Organising Committee for the advice and support. Special thanks go to the following people helping with the local organisation for all their efforts: the staff in the School's Academic Support Office, especially Ruth Maddocks, the staff of the School's Finance Office, the staff of the University who set up the registration and accommodation booking websites, as well as Rebekah Carter, Patrick Koopmann, Dmitry Tishkovski and Michał Zawidzki. Two people who deserve special mention are Mohammad Khodadadi for creating and maintaining the ARW website, and Fabio Papacchini for organising the reviewing of the abstracts, doing a lot of the reviewing himself and producing these proceedings. Moreover, I am very grateful to Ian Pratt-Hartmann, Chair of BCTCS 2012, and the BCTCS Steering Committee for agreeing to the collocation, and Ian and his team of helpers for the shared effort in organising both events.

Finally, it is my pleasure to acknowledge the generous support from the following organisations: the Automated Reasoning Workshop, the British Colloquium for Theoretical Computer Science, the British Logic Colloquium, the School of Computer Science at the University of Manchester, and the University of Manchester.

Manchester, April 2012

Renate Schmidt
ARW 2012 Chair

Workshop Organisation

Organiser

Renate A. Schmidt

ARW Organising Committee

Alexander Bolotov
Simon Colton
David Crocker
Louise Dennis
Clare Dixon
Jacques Fleuriot
Ullrich Hustadt
Mateja Jamnik
Ekaterina Komendantskaya
Alice Miller
Renate A. Schmidt
Volker Sorge

Local Organisation Team

Renate A. Schmidt
Mohammad Khodadadi
Fabio Papacchini
Dmitry Tishkovsky
Michał Zawidzki
Rebekah Carter
Patrick Koopmann
Ruth Maddocks

Sponsors

Automated Reasoning Workshop
British Colloquium for Theoretical Computer Science
British Logic Colloquium
School of Computer Science, The University of Manchester
The University of Manchester

Monday 2nd April 2012

14:00 Registration, tea and coffee — **Kilburn, lower first floor**

15:45 Welcome — **Kilburn, 1.1**

16:00 BCTCS/ARW invited lecture — **Kilburn, 1.1**

Mike Edmunds: The Antikythera Mechanism and the early history of mechanical computing

17:30 Reception — **Christie Library**

Tuesday 3rd April 2012

09:00 Registration — **Kilburn, lower first floor**

09:30 BCTCS/ARW invited lecture — **Kilburn, 1.1**

Reiner Hähnle: Formal verification of software product families

10:30 Tea & coffee

11:00 Parallel sessions:

BCTCS contributed talks A — **Kilburn, 1.1**

Phillip James: Domain-specific languages and automatic verification

Richard Barraclough: A unifying theory of control dependence and its application to arbitrary program structures

Patrick Totzke: Weak bisimulation approximants for BP processes

BCTCS contributed talks B — **Kilburn, 1.4**

Giles Reger: Quantified event automata: towards expressive and efficient runtime monitors

Andrew Lawrence, Ulrich Berger: Extracting a DPLL Algorithm

David Love: Why Don't Cantor's Sets Compute?

ARW short talks 1 (5 minutes each) — **Kilburn, 1.5**

C. Nalon: A Linear Strategy for Modal Resolution

A. Niknafs Kermani, B. Konev: Symmetry Theorem Proving

R. Williams, B. Konev: Simplified Temporal Resolution Using SAT Solvers

C. Sticksel, K. Korovin: A Note on Model Representation and Proof Extraction in the First-Order Instantiation-based Calculus Inst-Gen

M. Meri: Inverse Resolution in Case-Based Planning Cycle

A. Bolotov, V. Shangin: Natural Deduction in the Setting of Paraconsistent and Paracomplete Logics PCont and PComp

followed by poster session 1 at 11:30 — **Kilburn, lower first floor**

12:30 Buffet lunch

13:30 Parallel sessions:

BCTCS invited lecture **Kilburn, 1.1**

Nicole Schweikardt: On the expressive power of logics with invariant uses of arithmetic predicates

ARW short talks 2 (5 minutes each) **Kilburn, 1.5**

B. Lellmann, D. Pattinson: Graphical Construction of Cut Free Sequent Systems Suitable for Backwards Proof Search

Ş. Minică: Automatic Reasoning for Interrogative Logics

M. Zawidzki: Terminating Tableau Calculus for the Logic $K(E_n)$

D. Tishkovsky, C. Dixon, B. Konev, R. A. Schmidt: A Labelled Tableau Approach for Temporal Logic with Constraints

F. Papacchini, R. A. Schmidt: Minimal Models for Modal Logics

M. J. Gabbay (with C. Wirth): Quantifier Rules in Reductive Proof Using Nominal Semantics

followed by poster session 2 at 14:00 **Kilburn, lower first floor**

14:30 Parallel sessions:

BCTCS contributed talk A **Kilburn, 1.1**

Robert Piro: Model-theoretic characterisation of TBoxes and the TBox rewritability Problem

BCTCS contributed talk B **Kilburn, 1.4**

Christopher Thompson-Walsh: Extending a Rule-Based Biological Modelling Language Semantics with Containment

15:00 Tea & coffee

15:30 Parallel sessions:

BCTCS contributed talks A **Kilburn, 1.1**

Stanislaw Kikot: The length of query rewriting for OWL 2 QL

Paolo Guagliardo: On the relationship between view updates and logical definability

Chiara Del Vescovo: The modular structure of an ontology: atomic decomposition

BCTCS contributed talks B **Kilburn, 1.4**

Michael Gabbay: A very simple, explicit construction of some models of beta-equality and beta-eta-equality

Tie Hou: Modeling a language of realizers using domain-theoretic semantics

Hugh Steele: Double glueing and MLL full completeness

ARW short talks 3 (5 minutes each) **Kilburn, 1.5**

R. Carter, E. M. Navarro-López: Model Checking by Abstraction for Proving Liveness Properties of Hybrid Dynamical Systems

W. Denman: Verification of Nonpolynomial Systems Using MetiTarski

A. Piel: A Formal Behaviour Representation for the Analysis of Distributed Simulations of Unmanned Aircraft Systems

M. Alzahrani, L. Georgieva: Analysing Data-Sensitive and Time-Sensitive Web Applications

Y. Lu, A. Miller: Timed Analysis of RFID Distance Bounding Protocols

R. Kirwan, A. Miller: Progress on Model Checking Robot Behaviour

followed by poster session 3 at 16:00 **Kilburn, lower first floor**

17:00 ARW Business meeting (organisation committee) **Kilburn, LF15**

19:30 ARW dinner **Piccolino Ristorante e Bar**

Wednesday 4th April 2012

09:00 BCTCS/ARW invited talk **Kilburn, 1.1**

Daniel Kroening: SAT over an Abstract Domain

10:00 Parallel sessions:

BCTCS contributed talk A **Kilburn, 1.1**

Jian Song: 4-coloring H-Free Graphs when H is small

BCTCS contributed talk B **Kilburn, 1.4**

Martin Sticht: A Game-Theoretic Decision Procedure for the constructive Description Logic $c\mathcal{ALC}$

ARW short talks 4 (5 minutes each) **Kilburn, 1.5**

W. Sonnex: Deforestation + Dependent Types = Automated Induction

M. Brain: Using Algebra to Understand Search Spaces

A. Armstrong, G. Struth: Automated Reasoning in Higher-Order Algebra

Q. Mahesar, V. Sorge: Generation of Large Size Quasigroup Structures Using Algebraic Constraints

O. Al-Hassani, Q. Mahesar, C. Sacerdoti Coen, V. Sorge: A Term Rewriting System for Kuratowski's Closure-Complement

F. Cavallo, S. Colton, A. Pease: Uncertainty Modelling in Automated Concept Formation

M. Khodadadi, D. Tishkovsky, R. A. Schmidt: METTEL^2: Towards a Prover Generation Platform

10:30 Tea & coffee

11:00 Parallel sessions:

BCTCS contributed talks A **Kilburn, 1.1**

Jude-Thaddeus Ojiaku: Online makespan scheduling of linear deteriorating jobs on parallel machines

Tom Grant: Maximising lifetime for fault-tolerant target coverage in sensor networks

Mihai Burcea: Online multi-dimensional dynamic bin packing of unit fraction and power fraction items

BCTCS contributed talks B **Kilburn, 1.4**

Christopher Hampson: Modal Products with the difference operator

Brandon Bennett: An 'almost analytic' sequent calculus for first-order S5 with constant domains

Chris Banks: Towards a logic of biochemical processes

ARW poster session 4 **Kilburn, lower first floor**

12:30 Buffet lunch

13:30 LMS invited lecture in discrete mathematics **Kilburn, 1.1**

Rod Downey: Fundamentals of Parameterized Complexity I

14:30 Parallel sessions:

BCTCS contributed talk A **Kilburn, 1.1**

Yavor Nenov: Computability of topological logics over Euclidean spaces

BCTCS contributed talk B **Kilburn, 1.4**

Evelyn-Denham Coates: Optimum sort algorithms with o(N) moves

15:00 Tea & coffee

15:30 Parallel sessions:

BCTCS contributed talks A **Kilburn, 1.1**

Martin Adamčík: Collective reasoning under uncertainty and inconsistency

Murdoch Gabbay: Game semantics using nominal techniques

Arnoud Pastink: Approximate Nash Equilibria in an uncoupled setup with limited communication

BCTCS contributed talks B **Kilburn, 1.4**

Thomas Gorry: Communication-less agent location discovery

Dirk Sudholt: The analysis of evolutionary algorithms: why evolution is faster with crossover

Yanti Rusmawati: Dynamic networks as concurrent systems and supervised evolution

17:00 BCTCS committee meeting **Kilburn, LF15**

19:30 BCTCS dinner **Little Yang Sing**

Thursday 5th April 2012

09:00 LMS invited lecture in discrete mathematics **Kilburn, 1.1**

Rod Downey: Fundamentals of Parameterized Complexity II

10:00 BCTCS contributed talk **Kilburn, 1.1**

Sam Jones: Groups, formal language theory and decidability

10:30 Tea & coffee

11:00 BCTCS contributed talks **Kilburn, 1.1**

Domagoj Vrgoc: Regular expressions for data words

Laurence Day: The Silence of the Lambdas

Alistair Stewart: Polynomial time algorithms for multi-type branching processes and stochastic context-free grammars

12:30 Buffet lunch

13:30 Close

Table of Contents

The Antikythera Mechanism and the Early History of Mechanical Computing

Mike Edmunds

School of Physics and Astronomy, University of Cardiff

Abstract: Perhaps the most extraordinary surviving relic from the ancient Greek world is a device containing over thirty gear wheels dating from the late 2nd century B.C., and now known as the Antikythera Mechanism. This device is an order of magnitude more complicated than any surviving mechanism from the following millennium, and there is no known precursor. It is clear from its structure and inscriptions that its purpose was astronomical, including eclipse prediction. In this illustrated talk, I will outline the results - including an assessment of the accuracy of the device - from our international research team, which has been using the most modern imaging methods to probe the device and its inscriptions. Our results show the extraordinary sophistication of the Mechanism's design. There are fundamental implications for the development of Greek astronomy, philosophy and technology. The subsequent history of mechanical computation will be briefly sketched, emphasising both triumphs and lost opportunities. . . . . .

Formal Verification of Software Product Families

Reiner Hähnle

Technische Universitaet Darmstadt

Abstract: Formal verification techniques for software product families not only analyse individual programs, but act on the artifacts and components which are reused to obtain multiple software products. As the number of products is exponential in the number of artifacts, it is essential to perform verification in a modular fashion instead of verifying each product separately: the goal is to reuse not merely software artifacts, but also their verification proofs. In our setting, we realize code reuse by delta-oriented programming, an approach where a core program is gradually transformed by code deltas each of which corresponds to a product feature. The delta-oriented paradigm is then extended to contract-based formal specifications and to verification proofs. As a next step towards modular verification we transpose Liskovs behavioural subtyping principle to the delta world. Finally, based on the resulting theory, we perform a syntactic analysis of contract deltas that permits us to automatically factor out those parts of a verification proof that stays valid after applying a code delta.

A Linear Strategy for Modal Resolution

Cláudia Nalon

Departamento de Ciência da Computação – Universidade de Brasília
Caixa Postal 4466 – Brasília - DF - Brazil, CEP: 70.910-090
nalon@unb.br

Abstract: We discuss ongoing work on strategies for clausal resolution methods for normal modal logics. We propose to add new inference rules to existing resolution based-method for the mono-modal system K, so that a linear strategy can be applied to reduce the search space for a proof.

1 Introduction

Modal logics have been used to describe a variety of complex computational systems. Once the system is described by means of a logical language, an automated tool (e.g. a theorem prover) can be used to reason about the desired properties of that system.

In [4], we have presented resolution-based methods for several propositional normal modal logics, that is, where the schema $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$ (the axiom **K**), where φ and ψ are well-formed formulae, is valid. The methods presented in [4] deal with logics based on **K** and combinations of the axioms **T** ($\Box\varphi \Rightarrow \varphi$), **D** ($\Box\varphi \Rightarrow \Diamond\varphi$), **4** ($\Box\varphi \Rightarrow \Box\Box\varphi$), **5** ($\Diamond\varphi \Rightarrow \Box\Diamond\varphi$), and **B** ($\Diamond\Box\varphi \Rightarrow \varphi$). These methods are sound, complete, and terminating.

It is well known that proof methods for propositional logics are intractable [1]. However, the use of strategies for resolution-based methods can lead to spatial-efficient methods [5]. It is also well known that the satisfiability problem for K, the mono-modal logic based on the axiom **K**, is PSPACE [2]. Here we investigate a linear strategy for the resolution proof method for K [4]. In the next section, we present the syntax and semantics of K. In Section 3, we present the normal form and the resolution rules for K.

2 The Normal Logic K

The normal modal system K is an extension of the classical propositional logic with the operator $\Box$, where the axiom **K** holds. Formulae are constructed from a denumerable set of **propositional symbols**, $\mathcal{P} = \{p, q, p', q', p_1, q_1, \ldots\}$. For the mono-modal case, besides the propositional connectives ($\neg, \wedge$), we introduce an unary modal operator $\Box$, where $\Box\varphi$ is read as "the agent considers φ necessary". The fact that an agent considers φ possible, i.e. $\Diamond\varphi$, is denoted by $\neg\Box\neg\varphi$. The set of well-formed formulae, $\mathsf{WFF}_{\mathsf{K}}$, is defined in the usual way: the propositional symbols are in $\mathsf{WFF}_{\mathsf{K}}$; **true** is in $\mathsf{WFF}_{\mathsf{K}}$; if φ and ψ are in $\mathsf{WFF}_{\mathsf{K}}$, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, and $\Box\varphi$. A **literal** is either a proposition or its negation. $\mathcal{L}$ is the set of literals. A **modal literal** is either $\Box l$ or $\neg\Box l$, where $l \in \mathcal{L}$.

A **Kripke structure** M **over** $\mathcal{P}$ is a tuple $M = \langle \mathcal{S}, \pi, \mathcal{R} \rangle$, where $\mathcal{S}$ is a set of possible *worlds* (or *states*) with a distinguished world s_0; the function $\pi(s) : \mathcal{P} \rightarrow \{true, false\}$, $s \in \mathcal{S}$, is an interpretation that associates with each state in $\mathcal{S}$ a truth assignment to propositions; and $\mathcal{R} \subseteq S \times S$ is a binary relation on $\mathcal{S}$.

The binary relation $\mathcal{R}$ captures the possibility relation according to the agent: a pair (s, t) is in $\mathcal{R}$ if the agent considers world t possible, given her information in world s. We write $(M, s) \models \varphi$ to say that φ is true at world s in the Kripke structure M. Truth of classical formulae are given as usual; for modal formulae, we have that $(M, s) \models \Box\varphi$ iff for all t, such that $(s, t) \in \mathcal{R}$, $(M, t) \models \varphi$. The formulae **false**, $(\varphi \vee \psi)$, and $(\varphi \Rightarrow \psi)$ are introduced as the usual abbreviations for $\neg$**true**, $\neg(\neg\varphi \wedge \neg\psi)$, and $(\neg\varphi \vee \psi)$, respectively. Formulae are interpreted with respect to the distinguished world s_0. Let $M = \langle \mathcal{S}, \pi, \mathcal{R} \rangle$ be a Kripke structure with a distinguished world s_0. A formula φ is said to be **satisfiable in M** if $(M, s_0) \models \varphi$; φ is said to be **satisfiable** if there is a model M such that $(M, s_0) \models \varphi$; and φ is said to be **valid** if for all models M, $(M, s_0) \models \varphi$.

3 The Resolution Method for K

Formulae in the language of K can be transformed into a Separated Normal Form for Normal Logics (SNF_K). As satisfiability is defined in terms of the distinguished world s_0, we introduce a nullary connective **start**, where $(M, s) \models$ **start** iff $s = s_0$. A formula in SNF_K is represented by a conjunction of clauses, which are true at all reachable states, that is, they have the general form $\Box^* \bigwedge_i A_i$ where A_i is a clause, where $\Box^*$ is the *universal* operator. The formula $\Box^*\varphi$ holds iff φ holds at the actual world and at all reachable worlds, where reachability is defined in the usual way. The universal operator ensures that the *translation* of a formula is true at the actual world and at all reachable worlds. A clause may be **initial** (**start** $\Rightarrow \bigvee_{b=1}^{r} l_b$), **literal** (**true** $\Rightarrow \bigvee_{b=1}^{r} l_b$), **positive modal** ($l' \Rightarrow \Box l$), or **negative modal** ($l' \Rightarrow \neg\Box l$), where $l, l', l_b \in \mathcal{L}$. Transformation rules and their correctness can be found in [4].

Once a formula has been transformed in its normal form, the resolution method can be applied. In the following, l, l', l_i, $l'_i \in \mathcal{L}$ ($i \in \mathbb{N}$) and D, D' are disjunctions of literals. The inference rules are shown in Figure 1.

An initial clause may be resolved with either a literal clause or an initial clause (IRES1 and IRES2). Literal clauses can be resolved together (LRES). MRES is also

$$\text{[IRES1]}\quad \begin{array}{lcl} \Box^*(\textbf{true} & \Rightarrow & D \lor l) \\ \Box^*(\textbf{start} & \Rightarrow & D' \lor \neg l) \\ \hline \Box^*(\textbf{start} & \Rightarrow & D \lor D') \end{array} \qquad \text{[IRES2]}\quad \begin{array}{lcl} \Box^*(\textbf{start} & \Rightarrow & D \lor l) \\ \Box^*(\textbf{start} & \Rightarrow & D' \lor \neg l) \\ \hline \Box^*(\textbf{start} & \Rightarrow & D \lor D') \end{array}$$

$$\text{[LRES]}\quad \begin{array}{lcl} \Box^*(\textbf{true} & \Rightarrow & D \lor l) \\ \Box^*(\textbf{true} & \Rightarrow & D' \lor \neg l) \\ \hline \Box^*(\textbf{true} & \Rightarrow & D \lor D') \end{array} \qquad \text{[MRES]}\quad \begin{array}{lcl} \Box^*(l_1 & \Rightarrow & \Box l) \\ \Box^*(l_2 & \Rightarrow & \neg\Box l) \\ \hline \Box^*(\textbf{true} & \Rightarrow & \neg l_1 \lor \neg l_2) \end{array}$$

$$\text{[GEN1]}\quad \begin{array}{c} \Box^*(l'_1 \Rightarrow \Box\neg l_1) \\ \vdots \\ \Box^*(l'_m \Rightarrow \Box\neg l_m) \\ \Box^*(l' \Rightarrow \neg\Box\neg l) \\ \Box^*(\textbf{true} \Rightarrow l_1 \lor \ldots \lor l_m \lor \neg l) \\ \hline \Box^*(\textbf{true} \Rightarrow \neg l'_1 \lor \ldots \lor \neg l'_m \lor \neg l') \end{array} \qquad \text{[GEN2]}\quad \begin{array}{c} \Box^*(l'_1 \Rightarrow \Box l_1) \\ \Box^*(l'_2 \Rightarrow \Box\neg l_1) \\ \Box^*(l'_3 \Rightarrow \neg\Box\neg l_2) \\ \hline \Box^*(\textbf{true} \Rightarrow \neg l'_1 \lor \neg l'_2 \lor \neg l'_3) \end{array} \qquad \text{[GEN3]}\quad \begin{array}{c} \Box^*(l'_1 \Rightarrow \Box\neg l_1) \\ \vdots \\ \Box^*(l'_m \Rightarrow \Box\neg l_m) \\ \Box^*(l' \Rightarrow \neg\Box\neg l) \\ \Box^*(\textbf{true} \Rightarrow l_1 \lor \ldots \lor l_m) \\ \hline \Box^*(\textbf{true} \Rightarrow \neg l'_1 \lor \ldots \lor \neg l'_m \lor \neg l') \end{array}$$

Figure 1: Inference Rules for K

equivalent to classical resolution, as a formula and its negation cannot be true at the same state. The GEN1 rule corresponds to generalisation and several applications of classical resolution. GEN2 is a special case of GEN1. GEN3 is similar to GEN1 but the contradiction occurs between the right hand side of the positive i-clauses and the literal clause. We assume standard simplification from classical logic to keep the clauses as simple as possible.

4 A Linear Strategy

Linear resolution [3] is a popular strategy for reducing the search space for a proof in resolution-based methods. A **linear derivation** from a set of clauses $\mathcal{S}$ is a list of clauses $(C_1 \ldots, C_n)$, where $C_1 \in \mathcal{S}$ and each C_{i+1} is the resolvent of C_i and B, where either $B \in \mathcal{S}$ or $B \in \{C_1, \ldots, C_{i-1}\}$. If C_n is the empty clause, then $(C_1 \ldots, C_n)$ is a **linear refutation**.

The inference rules shown in Section 3 are not enough to produce linear refutations in the modal case. For instance, there is no linear refutation for the following set of clauses

$$\begin{array}{rrcl} 1. & \textbf{start} & \Rightarrow & a \lor b \\ 2. & a & \Rightarrow & \neg\Box\neg c \\ 3. & b & \Rightarrow & \neg\Box\neg c \\ 4. & \textbf{true} & \Rightarrow & \neg c \end{array}$$

although the set is unsatisfiable. We propose to extend the set of rules shown in Figure 1 with the following inference rules

$$\text{[LLHR]}\quad \begin{array}{lcl} \Box^*(\textbf{true} & \Rightarrow & l_1 \lor \ldots \lor l_n) \\ \Box^*(l_1 & \Rightarrow & l'_1) \\ & \vdots & \\ \Box^*(l_n & \Rightarrow & l'_n) \\ \hline \Box^*(\textbf{true} & \Rightarrow & l'_1) \\ & \vdots & \\ \Box^*(\textbf{true} & \Rightarrow & l'_n) \end{array}$$

where l_i, l'_i are *modal literals*. Similar inference rules are also needed for dealing with initial clauses. These rules are obviously sound. Our next step is to prove that by adding these inference rules the strategy is complete, i.e,

Theorem 4.1 *Let $\mathcal{S}$ be a set of* SNF_K *clauses. There is a refutation for $\mathcal{S}$ if, and only if, there is a linear refutation for $\mathcal{S}$.*

Acknowledgements

This work was partially funded by CAPES Foundation (Bolsista BEX 8712/11-5).

References

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[2] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[3] D. W. Loveland. A Linear Format for Resolution. In *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 147–162, Berlin, 1970. Springer-Verlag.

[4] Cláudia Nalon and Clare Dixon. Clausal resolution for normal modal logics. *J. Algorithms*, 62:117–134, July 2007.

[5] Jacobo Torán. Space and width in propositional resolution (column: Computational complexity). *Bulletin of the EATCS*, 83:86–104, 2004.

Symmetry Temporal Theorem Proving

Amir Niknafs kermani[1] Boris Konev[2] Michael Fisher[3]

[1] The University of Liverpool, amir.niknafs.kermani@gmail.com
[2] The University of Liverpool, konev@liv.ac.uk
[3] The University of Liverpool, mfisher@liv.ac.uk

Abstract: In this paper we consider the verification of Propositional Temporal Logic in symmetric systems. In particular, we try to verfiy a specification of a system with few number of processes and use the outcome in order to reduce the verification complexity of the same system with larger number of processes.

1 Introduction

Propositional Temporal Logic(PTL) [4] is an extension of classical propositional logic, specifically adding operators relating to time. In addition to classical logic, temporal logics often contain operators such as $\bigcirc$, the next moment in time; $\square$, always in the future; and $\diamond$, sometime in the future.

If we have specified a system using the propositional temporal formula ψ, then we can check whether a certain temporal property φ, follows from the specification of the system by establishing that φ is implied by ψ. This can be done using Temporal Logic theorem proving [4]. Currently there are several automated provers such as *TeMP* [7] and *TSPASS* [8]. Alternatively to theorem provigin for PTL, one can use *Temporal Logic Model Checking* to prove some properties within a system. e.g. Model checkers such *SPIN Model Checker* [6] have proved to successfully use symmetry properties of a system to increase efficiency; this was an inspiration for our work.

The core aim of this research is to improve performance for PTL theorem proving in symmetric systems. These systems consist of some identical tasks which can be the same for all the involved processes, resource sharing i.e. *Cache Coherence Protocol* [1] is a good example of a symmetric system. The use of local caches in multiprocessor systems reduce both memory access latency and network traffic, since each cache holds local copies of main memory blocks. However, one has to ensure that the copies of the same memory block in the caches of different processors are consistent for any given number of processes. Such data consistency can be provided by *Cache Coherence Protocol*(CCP). We apply a temporal prover to the system specification for Specific Number of processes to discover its symmetry.

2 Symmetry Theorem Prover

We want to improve the efficiency of Clausal Temporal Resolution [9] for symmetric systems. An overview of the Clausal Temporal Resolution method is as follows:

1. Transform formula A into *Seperate Normal Form*(SNF) [4], giving a set of clauses A_s.
2. Perform *step resolution* [4] on clauses from A_s until either:
   (a) A contradiction is derived, in which case A is unsatisfiable; or
   (b) No new resolvents are generated, in which case we continue to step(3).
3. Select an eventuality from within A_s, and perform temporal resolution [4] with respect to this - if any new formulae are generated, go back to step (2).
4. If all eventualities have been resolved, then A is satisfiable, otherwise go back to step (3).

The main concern with the above algorithm is that *Temporal Resolution* can be quite slow. There have been several attempts to increase its speed but it remains the slowest, yet necessary, part of the method [2, 3].
In order to be able to perform *Temporal Resolution* we require to search for *loops* [2]. Once we have found the loops we can carry on with the process using the step resolution. In this research we aim to reduce the time for loop search by using symmetry, the following algorithm aims to demonstrate this.

1. Run the Resolution Method for a symmetric system P with small number of processes i[1], possibly for P_i and P_{i+1} . (Use this in essentially a propositional temporal problem.)
2. From the results gathered from step one, try to guess a loop for a protocol for a larger number of processes.
3. If no Loop could be guessed then terminate(we were unable to determine the symmetry among processes).
4. Check if the guessed formula is indeed a loop for the specification otherwise go back to step 2 and try to guess another Loop.
5. Remove the eventuality clause and add the generated loop to the formula and run the Theorem Prover.

[1]the minimum number of required processes to prove a property.

Table 1: theorem prover comparison (Using Temp theorem prover)

Number of processes	Original Problem	Modified Problem	results
2	0.060s	0.011s	Unsatisfiable
3	1.240s	0.036s	Unsatisfiable
4	16.124s	0.134s	Unsatisfiable
5	119.662s	0.640s	Unsatisfiable
6	1717.886s	4.138s	Unsatisfiable
7	∞	35.108s	Unsatisfiable
8	∞	340.408s	Unsatisfiable
9	∞	4249.012s	Unsatisfiable

3 Experimental results

The algorithm on the previous section has been tested on the MSI Protocol [1]. We ran the theorem prover on MSI_2[2] and MSI_3[3] to aquire enough information to guess a satisfaying loop. Then using the Temp theorem prover [7] and has been compared to the results of prover on the original formula in the table 1. *Modified Problem* column from table 1 is the results of the using the clauses representing the loop instead of the loop itself.

4 Future Work

Currently this research aims at minimizing the time required to find a loop. However, as it appears on the experiments shows that for a larger number processes the provers still find it difficult to cope. Therefore, to further improve the efficiency we will investigate the possibilities of using SAT solver [5] in the algorithm.

References

[1] Giorgio Delzanno. Automatic verification of parameterized cache coherence protocols. In *Proceedings of the 12th International Conference on Computer Aided Verification*, CAV '00, pages 53–68, London, UK, 2000. Springer-Verlag.

[2] Clare Dixon. Temporal resolution using a breadth-first search algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[3] Carmen Fernandez-Gago, Michale Fisher, and Clare Dixon. An algorithm for guiding clausal temporal resolution. In *4th International Workshop on Strategies in Automated Deduction (STRATEGIES'01)*, Siena, Italy, June 2001.

[4] M. Fisher. *Practical Formal Methods Using Temporal Logic*. John Wiley & Sons, 2011.

[5] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Chapter 2 satisfiability solvers. In Vladimir Lifschitz Frank van Harmelen and Bruce Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 89 – 134. Elsevier, 2008.

[6] Gerard J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, september 2003.

[7] Ullrich Hustadt, Boris Konev, Alexandre Riazanov, and Andrei Voronkov. Temp: A temporal monodic prover. In *In Proc. IJCAR-04, LNAI*, pages 326–330. Springer, 2004.

[8] Michel Ludwig and Ullrich Hustadt. Implementing a fair monodic temporal logic prover. *AI Commun.*, 23:69–96, April 2010.

[9] Clare Dixon Michael Fisher and Martin Peim. Clausal temporal resolution. *ACM Trans. Comput. Logic*, 2:12–56, January 2001.

[2]Specification of MSI protocol with 2 processes.
[3]Specification of MSI protocol with 3 processes.

Simplified Temporal Resolution using SAT Solvers

Richard Williams Boris Konev

Department of Computer Science, University of Liverpool, Liverpool L69 7ZF
{R.M.Williams1, Konev}@liv.ac.uk

Abstract: There have been a number of implementations of clausal temporal resolution based on the well-known normal form (SNF) and resolution rules. This paper explores an implementation of a proposed, but never implemented, resolution method featuring a refined normal form (DSNF) and simplified resolution rules. The method reduces a large proportion of the problem to classic propositional logic facilitating the use of Boolean Satisfiability (SAT) solvers. We present a discussion of the implementation and the strategies it employs as well as the initial results of the comparative performance of the new solver against existing solutions.

1 Introduction

Propositional Linear-time Temporal Logic (PLTL) has been used in various areas of computer science, for example, for the specification and verification of distributed and concurrent systems [6] or verification via model checking [1]. Many applications require to know if a given PLTL formula is *(un)satisfiable*, which can be established with a number of techniques including automata-based approaches [11], tableau methods [12] and clausal temporal resolution [3]. Clausal temporal resolution has been successfully implemented [5, 4] and shown to perform well in practice [10].

Simplified temporal resolution introduced [2] builds on clausal temporal resolution introduced by Fisher [3] by adopting a relaxed notion of a normal form and much more abstractly described inference rules. The advantage of simplified temporal resolution is that it provides a cleaner separation between classical and temporal reasoning, ensures more streamlined use of temporal resolvents and requires only unconditional eventualities. In this paper we present an implementation of simplified temporal resolution using SAT solvers and evaluate the implementation on known benchmarks.

2 Preliminaries

Arbitrary PLTL-formulae can be transformed into *divided separated normal form* (DSNF) in a satisfiability equivalence preserving way using a renaming technique replacing non-atomic subformulae with new propositions and removing all occurrences of the $\mathcal{U}$ ('until') and $\mathcal{W}$ ('unless') operator [3, 2]. The result is a *DSNF problem* of the following form (where $\bigcirc$, $\Box$, and $\Diamond$ denote 'next', 'always', and 'eventually', respectively): a universal part, $\mathcal{U}$, given by a set of propositional formulas (clauses); an initial part, $\mathcal{I}$, with the same form as the universal part; a step part, $\mathcal{S}$, given by a set of propositional step temporal clauses of the form: $P \Rightarrow \bigcirc Q$ where P and Q are Boolean combinations of propositional symbols; and an eventuality part, $\mathcal{E}$, given by unconditional eventuality clauses of the form $\Diamond l$, where l is a literal. The intended meaning of a DSNF problem is given by $\mathcal{I} \wedge \Box\mathcal{U} \wedge \Box\mathcal{S} \wedge \Box\mathcal{E}$.

The inference system we use consists of an (implicit) *merging operation*

$$\frac{P_1 \Rightarrow \bigcirc Q_1, \ldots, P_n \Rightarrow \bigcirc Q_n}{\bigwedge_{j=1}^{n} P_i \Rightarrow \bigcirc \bigwedge_{j=1}^{n} Q_i},$$

and the following inference rules.

- *Step resolution rule*: $\dfrac{A \Rightarrow \bigcirc B}{\neg A}\ (\bigcirc^{\mathcal{U}}_{res})$, where $\mathcal{U} \cup \{B\} \vdash \bot$.
- *Sometime resolution rule*
$$\frac{A_1 \Rightarrow \bigcirc B_1, \quad \ldots, \quad A_n \Rightarrow \bigcirc B_n \qquad \Diamond l}{(\bigwedge_{i=1}^{n} \neg A_i)}\ (\Diamond^{\mathcal{U}}_{res}),$$
where $A_i \Rightarrow \bigcirc B_i$ are *merged* step rules such that $\mathcal{U} \cup \{B_i, l\} \vdash \bot$ and $\mathcal{U} \cup \{B_i, \bigwedge_{j=1}^{n} \neg A_j\} \vdash \bot$ for all i
- *Termination rules*
The contradiction $\bot$ is derived and the derivation is (successfully) terminated if $\mathcal{U} \cup \mathcal{I} \vdash \bot$, or if $\mathcal{U} \cup \{l\} \vdash \bot$, where l is an eventuality literal.

3 Implementation Details

As the side conditions of the inference rules are propositional problems containing no temporal operators, they can be tested with an external SAT Solver. All that remains is to find the appropriate merged step clause, or clauses, which satisfy the side conditions. For the set of all step clauses $\mathcal{S}$ its powerset $\mathcal{P}(\mathcal{S})$ represents the set of all possible merged clauses. We search through the elements of $\mathcal{P}(\mathcal{S})$ to find merged step clauses satisfying the rule side conditions. For example, in the case of step resolution the search procedure must return a node n such that $n \in \mathcal{P}(\mathcal{S})$ and n satisfies the step resolution side condition. Sometime resolution is significantly more complex as the procedure must find a set of merged clauses that satisfy much more complex side conditions. As a result the main loop in our solver prioritises step resolution.

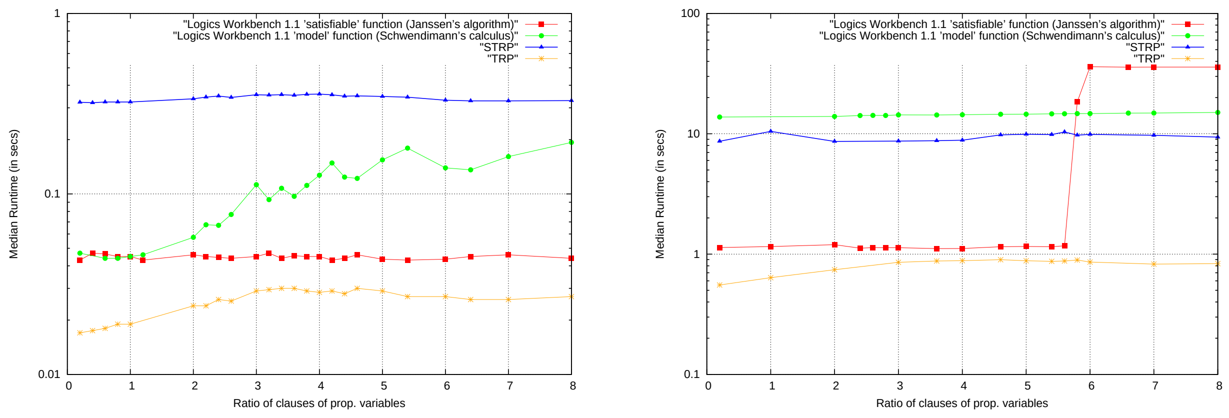


Figure 1: Results for C^1_{ran} (left) and C^2_{ran} (right)

3.1 Search Strategy

Our solver STRP (Simplified Temporal Resolution Prover) uses an A* search strategy [7]. The strategy makes use of a very simple heuristic function based on the intuition that the smallest (in terms of number of literals) clauses are more likely to generate a contradiction and thus satisfy the inference and termination side conditions. Thus the heuristic value of a given node n is defined as the number of propositional literals on the left-hand side (LHS) of the merged clause that the node n represents plus the number of literals on the right-hand side (RHS).

$$h(n) = \text{size(LHS)} + \text{size(RHS)}$$

4 Experiments

We conducted experiments on a two classes of semi-random benchmark formulae, C^1_{ran} and C^2_{ran} introduced in [5]. We use the following parameters for the random formulae $n = 5, k = 3$ and $p = 0.5$. The experiments involved STRP, a clausal resolution-based solver TRP [5] and two tableaux-based procedures implemented in Logics Workbench 1.1 (The satisfiable function [8] and the model function [9]). All experiments were conducted on PC with an Intel Dual-Core E2180, 2.0 GHz processor, with 2GB of RAM running Fedora 16. For each individual test a time-limit of 300 seconds was set.

The results (Figure 1) show promising performance on problems in C^1_{ran}; indeed STRP shows a similar trend in runtime to that of TRP for problems in this set. Performance on C^2_{ran} is less favourable but still quite interesting; problems in C^2_{ran} contain a fixed number of step clauses which impose only trivial contradictions. This results in a large number of goal nodes being produced before termination which leads to the increased runtime. Future work will involve conducting further experiments on benchmarks from [10], as well as exploring other strategies/optimizations to improve performance on difficult classes of problems such as those in C^2_{ran}.

References

[1] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 1999.

[2] A. Degtyarev, M Fisher, and B. Konev. A simplified clausal resolution procedure for propositional linear-time temporal logic. In *Tableaux 2002, Proceedings*, volume 2381 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2002.

[3] M. Fisher. A resolution method for temporal logic. In *Proc. IJCAI'91*, pages 99–104. Morgan Kaufman, 1991.

[4] U. Hustadt and B. Konev. Trp++2.0: A temporal resolution prover. In *CADE'03*, volume 2741, pages 274–278. Springer, 2003.

[5] U. Hustadt and R. A. Schmidt. Scientific benchmarking with temporal logic decision procedures. In *KR'02*, pages 533–546. Morgan Kaufmann, 2002.

[6] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.

[7] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 2010.

[8] Janssen, G. *Logics for Digital Circuit Verification: Theory, Algorithms, and Applications*. PhD thesis, Eindhoven University of Technology, The Netherlands. Springer, 1999.

[9] Schwendimann, S. *Aspects of Computational Logic*. PhD thesis, Universität Bern, Switzerland, 1998.

[10] V. Schuppan and L. Darmawan. Evaluating LTL satisfiability solvers. In *ATVA*, volume 6996 of *LNCS*. Springer, 2011.

[11] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32(2):183–219, 1986.

[12] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–152, 1985.

A Note on Model Representation and Proof Extraction in the First-Order Instantiation-based Calculus Inst-Gen

Konstantin Korovin
korovin@cs.man.ac.uk

Christoph Sticksel
sticksel@cs.man.ac.uk

School of Computer Science, The University of Manchester

Abstract: We describe the recent extensions of the instantiation-based theorem prover iProver to generate models of satisfiable and proofs for unsatisfiable inputs, both features being demanded by applications [1, 2].

1 Introduction

The main idea behind instantiation-based methods for first-order logic is to combine efficient ground reasoning with smart first-order instantiations. We have been developing the Inst-Gen calculus [4] and its equational variant Inst-Gen-Eq [6], where the ground reasoning is decoupled from instantiation, thus allowing one to employ efficient off-the-shelf SAT and SMT solvers for ground reasoning. This is unlike traditional calculi for first-order logic such as resolution or superposition, where first-order reasoning is tackled purely by applying first-order inference rules.

We focus on two aspects important to applications and users of automated theorem provers, namely the output of models and proofs. In the Inst-Gen method models can be extracted from a saturation of the input and we discuss how to obtain compact model representations for satisfiable input. Since the ground reasoning is delegated to a black-boxed solver, the extraction of proofs of unsatisfiability relies on the ground solver and we demonstrate how to use unsatisfiable cores from the ground solver in proof extraction without a loss of performance during the proof procedure.

2 The Inst-Gen Method

The basic idea of the Inst-Gen method is as follows. The input set of first-order clauses S is abstracted to a set of ground clauses $S\bot$ by mapping all variables to the same ground term, conventionally named $\bot$. If this ground abstraction is unsatisfiable, then the set of first-order clauses is also unsatisfiable. Otherwise, there is a ground model $I_\bot$ for the abstraction that is used to guide an instantiation process. The ground satisfiability check and construction of a ground model is delegated to a solver for satisfiability modulo theories (SMT) in the presence of equations or to a propositional (SAT) solver if no equational reasoning is required.

The ground model $I_\bot$ obtained from the solver 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. Inst-Gen is therefore composed of two parts: *ground satisfiability solving* on the abstraction of the set of clauses and *first-order reasoning on literals* corresponding to ground literals in the model of the abstraction.

The first-order instantiation process is guided by means of a selection function based on the ground model $I_\bot$. The *selection function* sel assigns to each first-order clause C in S exactly one literal $\mathrm{sel}(C) = L$ from C such that $I_\bot \models L\bot$. At least one such literal always exists as the ground abstraction of the clause is true in the model $I_\bot$.

We also employ a constraint mechanism for redundancy elimination that becomes crucial in model construction. Intuitively, a clause C represents all its ground instances, hence there are ground instances represented by both the clause C and an instance $C\sigma$ of it. In order to eliminate this duplication we attach a constraint to each clause, effectively blocking all ground instances of clause C that are represented by the instance $C\sigma$. We view such a *dismatching constraint* Φ as a set of substitutions $\{\tau_1, \dots, \tau_n\}$ and say that a substitution σ *satisfies* the dismatching constraint Φ if it is not more specific than any $\tau_i \in \Phi$.

For simplicity we only consider the non-equational calculus Inst-Gen, the results about model construction and proof extraction apply with some modification also to the equational calculus Inst-Gen-Eq. The following inference rule is applied to the input clause set up to saturation.

Inst-Gen inference rule

$$\frac{C \vee L \mid \Phi \quad D \vee \overline{L'} \mid \Psi}{(C \vee L)\,\sigma \quad \left(D \vee \overline{L'}\right)\sigma}$$

(i) $\sigma = \mathrm{mgu}(L, L')$, (iii) $\mathrm{sel}(D \vee \overline{L'}) = \overline{L'}$,
(ii) $\mathrm{sel}(C \vee L) = L$, (iv) σ satisfies Φ and Ψ.

Both clause instances $(C \vee L)\,\sigma$ and $\left(D \vee \overline{L'}\right)\sigma$ are added to the clause set and the dismatching constraints of the premises are extended to $\Phi \cup \{\sigma\}$ and $\Psi \cup \{\sigma\}$.

The Inst-Gen inference rule is similar to the Resolution inference rule, but instead of resolving away the complementary unifiable literals L and $\overline{L'}$ and combining the two premises into the new clause $(C \vee D)\,\sigma$, the clauses are instantiated with the most general unifier σ. We view a literal L as representing all its ground instances, hence having both the literal L and a unifiable complement $\overline{L'}$ selected means that there are ground instances $L\sigma$ and $\overline{L'}\sigma$ represented that are contradictory. The ground model, that only contains the abstractions $L\bot$ and $\overline{L'}\bot$, therefore cannot be extended to a first-order model due to this conflict between the first-order literals L and $\overline{L'}$. After adding the clause instances with the mgu σ the ground solver can witness this conflict and evolve the model of the ground abstraction appropriately. On the first-order level the dismatching constraint on the premises is extended with σ.

The Inst-Gen method is refutationally complete, that is, from an unsatisfiable input an exhaustive application of the inference rule will eventually lead to an unsatisfiable ground abstraction. On the other hand, if the clause set becomes saturated under the inference rule, it is satisfiable and we can extract a model.

3 Model Representation

If the set of clauses is closed under Inst-Gen inferences and the ground abstraction is satisfiable, then there is no first-order conflict on selected literals and the set of selected literals can indeed be interpreted as a model for the input clause set.

Instantiation-based methods generate a certain class of models that is commonly described with a *disjunction of implicit generalisation (DIG)* [3]. Instead of representing Inst-Gen models as DIGs we represent them as predicate definitions over the term algebra.

For each literal $(\neg)P(t_1,\ldots,t_n) \in \operatorname{sel}(S)$ that contains the variables $\bar{x} = \langle x_1,\ldots,x_m\rangle$ and occurs in a constrained clause $C \mid \Phi$ with $\Phi = \{\tau_1,\ldots,\tau_k\}$ we define

$$\begin{aligned}
&\Lambda_{(\neg)P(t_1,\ldots,t_n)}(y_1,\ldots,y_n) \rightleftharpoons \\
&\quad \exists\bar{x}[\, y_1 = t_1 \wedge \cdots \wedge y_n = t_n \wedge \\
&\qquad \forall\bar{z}_1(\, x_1 \neq x_1\tau_1 \vee \cdots \vee x_m \neq x_n\tau_n) \wedge \cdots \wedge \\
&\qquad \forall\bar{z}_k(\, x_1 \neq x_1\tau_k \vee \cdots \vee x_m \neq x_n\tau_k)],
\end{aligned}$$

where $\bar{y} = \langle y_1,\ldots,y_n\rangle$ is a tuple of fresh variables and $\operatorname{var}(\operatorname{rng}(\tau_i)) \subseteq \bar{z}_i$ (here $\bar{z}_i \cap \bar{x} = \emptyset$). The first conjunct containing the existential quantifier corresponds to a flattening of the atom $P(t_1,\ldots,t_n)$ into $P(y_1,\ldots,y_n)$, the remaining conjuncts express the dismatching constraints.

Given an Inst-Gen saturated set of clauses S we can extract several models. For each predicate we can collect either all *positive occurrences* or dually all *negative occurrences* and define the predicate in the model as

$$(\neg)P(y_1,\ldots,y_n) \iff \bigvee_{(\neg)P(t_1,\ldots,t_n)\in\operatorname{sel}(S)} \Lambda_{(\neg)P(t_1,\ldots,t_n)}.$$

Since the sizes of positive and negative representations can be vastly different, for each atom we can chose the smallest. Alternatively we can use *implied definitions* of the form

$$(\neg)P(y_1,\ldots,y_n) \Leftarrow \bigvee_{(\neg)P(t_1,\ldots,t_n\in\operatorname{sel}(S)} \Lambda_{(\neg)P(t_1,\ldots,t_n)},$$

and allow completing the model arbitrarily if undefined.

All model representations have been implemented in the iProver system, such that the user can choose the one most fit for purpose. The implementation keeps both the set of active literals as well as the dismatching constraints compactly stored in discrimination trees, thus the model can be efficiently constructed.

4 Proof Extraction

As soon as the ground solver finds the ground abstraction unsatisfiable, the input clause set has been proved unsatisfiable. Since we regard the ground solver as a black box and proof extraction in SAT solving may cause a considerable degradation in performance, we content ourselves with unsatisfiable cores returned from the solver. Then, a proof of unsatisfiability is a sequence of first-order inferences up to a set of clauses that is propositionally unsatisfiable.

We record each first-order inference step in iProver and run two instances of the ground solver in parallel, both containing the ground abstraction of the current clause set. The first solver instance is frequently called to check satisfiability, the second instance is used only a posteriori to obtain an unsatisfiable core. For this purpose we add a unique literal to each clause in the second instance and assume the complement of each tracking literal. The falsified assumptions represent an unsatisfiable core that we can minimise.

After the first instance of the ground solver reports unsatisfiability of the ground abstraction, the second instance is invoked for the first time. We map the failed assumptions to their first-order clauses and recursively trace each inference back to premises which are input clauses. Since proof extraction is a separate step, the performance of the proof search is not affected.

The iProver system makes use of global propositional subsumption [5], which simplifies a clause with respect to the some grounding. Since justifying each simplification step during the proof search would result in a severe performance hit, we postpone this step until the actual proof extraction. If in tracing the proof tree a clause is encountered that was obtained by global propositional subsumption, the second instance of the ground solver is invoked once more to find the justification for the simplification, a set of clauses younger than the simplified clause. We then continue the search for input clauses from this set.

We note that for many applications proofs in this form are sufficient. In particular the Sledgehammer tool of the Isabelle system [1] uses automated theorem provers essentially to filter an input clause set for a first-order unsatisfiable core, hence it requires much less than a detailed proof and can work with the output generated here.

References

[1] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT Solvers. In *CADE 23*, pages 116–130, 2011.

[2] M. Emmer, Z. Khasidashvili, K. Korovin, and A. Voronkov. Encoding Industrial Hardware Verification Problems into Effectively Propositional Logic. In *FMCAD 2010*, 2010.

[3] C. G. Fermüller and R. Pichler. Model Representation via Contexts and Implicit Generalizations. In *CADE 20*, pages 409–423, 2005.

[4] H. Ganzinger and K. Korovin. New Directions in Instantiation-Based Theorem Proving. In *LICS 2003*, pages 55–64, 2003.

[5] K. Korovin. iProver - An Instantiation-Based Theorem Prover for First-Order Logic (System Description). In *IJCAR 2008*, pages 292–298, 2008.

[6] K. Korovin and C. Sticksel. Labelled Unit Superposition Calculi for Instantiation-Based Reasoning. In *LPAR-17*, pages 459–473, 2010.

Inverse Resolution in Case-Based Planning Cycle

Martti Meri

Aalto University martti.meri@aalto.fi

Abstract: Conceptual process design is a preliminary stage of industrial process design, where model generation is the most characteristic mode of reasoning. The idea we present in this paper is an blend of reasoning modes and knowledge base structures supporting learning by inverse resolution using the case histories, and reasoning in forward direction using standard resolution to form a proof that acts as the frame for the conceptual process definition.

1 Introduction

Metallurgical industry uses processes that obey laws of physical, mineralogical, and chemical domain theories. This is one of the reasons conceptual process design, a phase of the process design, is a complicated engineering problem, mainly done based on expert intuition and practical experience. Decision support systems are required in the field and Case-Based Reasoning (CBR) is an auspicious candidate methodology [6]. The sub-domains are such that combining sub-symbolic, symbolic and ontological levels of representation are required for knowledge engineering. Conceptual process design is iterative by nature, following the classical phases of CBR-cycle, i.e. *retrieve*, *reuse*, *revise* and *retain*. Besides own internal knowledge bases there are case data available from research institutes and commercial sources. Inverse resolution has been studied in [4]. Earlier framework for integrating case-based reasoning and inductive learning is presented in [1]. In this paper we outline a new method based on logical theorem proving that uses case-based data to learn knowledge of behavior that has appeared in past solutions and modifies them to constitute to the proof of the current plan to be made. This new method integrating automated reasoning with CBR manages in parallel the current proof that essentially corresponds to the plan that is the solution to the planning problem given, and the case-base solutions (proofs) of case problems.

2 Knowledge

The type of integration requires well designed formal representation and for that purpose we use a kind of STRIPS-based action representation formalism. Actions name an operation and list their preconditions and effects. The aim at a logical theorem proving leads us to follow SATPLAN type of thinking, logical axioms for operators implying both their effects and preconditions. We assume that it is easier to gain access to the state fluents fluctuations along the history lines recorded by the cases. There states form a sequence and each state has some state fluents, i.e. positive or negative literals holding.

The case histories are arrays of arrays of attribute-value pairs, that is each $state_i$ is described as an array of attribute -value pairs, and states arrays form the history, see figure 1. Figure 1 represent the kinds of knowledge elements that

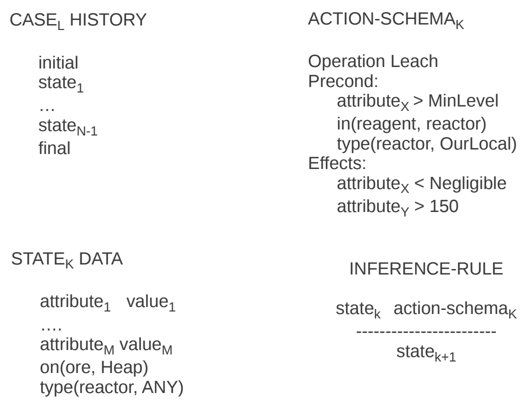


Figure 1: Elements of knowledge

are relevant and available in conceptual design of industrial processes. Once the conceptual process has been formed, it is possible to complete it using model generation methods that add the resources according to the domain ontologies and check the final model against various spatial and temporal molecular sub-theories. For the ontological part, feature-based similarity functions can be used. The promising thing is to retrieve work-flows that are similar than the candidate solution , as in [2].

3 Reasoning

The inference rule in the figure 1 can used in two ways: resolution and inverse resolution. We can deduce new states knowing current state and the action-schema, but we can also inductively form the action schema by looking at two consecutive states. When we have a set of cases that have been applied in a similar situation successfully and the action information for the first operation is similar we can collect the information about the preceding and following states and form our model of the operation based on this collected information. Once we have inductively formed actions that work for cases, we can try to use them in forward reasoning mode for solving new problems. A sequence of actions takes the system through a sequence of

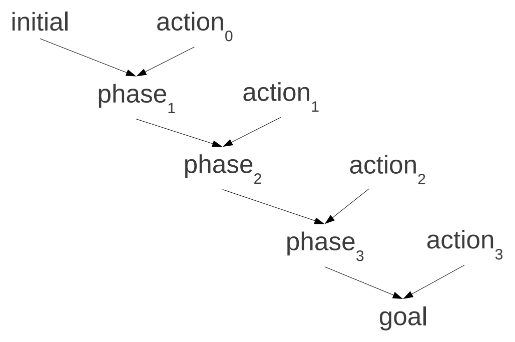


Figure 2: Process as a proof

states if we have a proof as presented in the figure 2. The solutions that are found are good cases as they are localized to our environment. Rule association mining can be used for handling the problem that some cases show exceptional behavior. Rules that have high support and confidence levels can be used as parts of the operation learning. In fact we see this as a possibility to statistically manage the classical ramification and qualification problems of AI-Planning. Fast implementation of the method will be based on efficient indexing that allows finding similar knowledge elements from the KB. Similarity, equality, conformance, matching are all relevant concept in CBP. Taxonomy for similarity concepts is provided by [3].

4 The method

Let's have a target case $\langle \Pi, P \rangle$, where Π is the planning problem initial and goal state description, a pair $\langle I, G \rangle$. The standard assumption is that we wish to imply partial order to the members of P. For the sake of simplicity and for the linearity of the proof we assume a total order of action to form the plan. Now our case-base is designed to consist of prior cases, source cases that have the same structure as our target case. The method here is to get in a target state's ${}^{i}S_T$ the n k-NN nearest neighbor states by comparing the I parts of the cases. Say these are ${}^{i}S_C^j, j \in [1, n]$. Now that we have located n similar past states in the case base we can look at what was done in these cases. The linear structure of the plans gives us the ${}^{i}Act_C^j$, which might or might not unify in the logical sense or conform to each other. The cases also tell us the goal reached and these goals are the original goal conditions of the planning problem, not the refined complete state descriptions. It is useful to retain the results of having followed the least commitment principle in the past planning efforts. The decision to be taken at this point includes two dimensions; we can look at the goal part and evaluate the cases based on the distance of the case goal to our target goal definition. Is that where we are aiming at, the other dimension being the action sequence of the case, for which the question is, do we have what it takes do perform this kind of action sequence. After all, the cases can be from other sources than our own past experience. Goal driven similarity assessment [5] makes similarity measures sensitive to the context of the search given by the goal, i.e. the G component in Π.

5 Conclusion

In this article we have presented some of the initial views of how this kind of conceptual engineering methodology could be constructed. Our method provides a novel view to see planning as theorem proving. A systematic knowledge engineering methodology for managing the distributed, multi-source knowledge and combining the reasoning systems to support the CBR-cycle at it's different phases would be invaluable. Naturally we can not do much about the worst case complexity that remains the same for both the generative and variant techniques, but recent advances in rule association mining and kernel functions as well as ontological support bringing in representation that remain guarded from the point of view of computational complexity promise a lot. The method as such is generalisable to other domains.

References

[1] Eric Auriol, Michel Manago, Klaus dieter Althoff, Stefan Wess, and Stefan Dittrich. Integrating induction and case-based reasoning: Methodological approach and first evaluations. In *Proc. 17th Conference of the GfKl*, pages 18–32. Springer Verlag, 1994.

[2] Ralph Bergmann and Yolanda Gil. Retrieval of semantic workflows with knowledge intensive similarity measures. In *ICCBR*, 2011.

[3] Padraig Cunningham. A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Trans. Knowl. Data Eng.*, 21(11):1532–1543, 2009.

[4] David Hume and Claude Sammut. Using inverse resolution to learn relations from experiments. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 412–416. Morgan Kaufmann, 1991.

[5] Dietmar Janetzko, Stefan Wess, and Erica Melis. Goal-driven similarity assessment. In *GWAI-92 16th German Workshop on Artificial Intelligence, volume 671 of Springer Lecture Notes on AI*, pages 283–298. Springer Verlag, 1992.

[6] Lotta Rintala, Jari Aromaa, and Olof Forsén. The use of decision methods in the selection of leaching alternatives. In *The 6th international European Metallurgical Conference*, 2011.

Natural Deduction in the setting of Paraconsistent and Paracomplete Logics PCont and PComp

Alexander Bolotov[1] Vasilyi Shangin[2] *

[1] University of Westminster, UK
A.Bolotov@wmin.ac.uk
[2] Moscow State University, Russia
shangin@philos.msu.ru

To $\boxed{\text{Vyacheslav Bocharov}}$ - our teacher and friend.

Introduction

In this paper we tackle the problem of constructing the natural deduction calculi and corresponding proof procedures for paraconsistent logic PCont [3] and paracomplete logic PComp [1]. The former logic is used, in particular, for reasoning about contradictory where paradoxes do not lead to the 'deductive explosion', i.e. where formulae of the type **false** $\supset A$, for any A, are not valid. The logic PComp is useful to reason about uncertain systems where the law of excluded middle does not hold. We complete the proof searching procedure for the logic PCont and formulate the natural deduction system for PComp.

A sound and complete natural deduction system NPCont and NPComp

We first introduce the semantics of these logics and then the natural deduction rules with the common base rules and with the characteristic rules for both.

We fix a standard propositional language L over an alphabet $p, q, r, p_1, q_1, r_1, \ldots$. The following is a set of base axioms for PCont and PComp.

1. $A \supset (A \vee B)$
2. $A \supset (B \vee A)$
3. $(A \wedge B) \supset A$
4. $(A \wedge B) \supset B$
5. $A \supset (B \supset A)$
6. $\neg\neg A \supset A$
7. $A \supset \neg\neg A$
8. $(A \supset B) \supset ((B \supset C) \supset (A \supset C))$
9. $(C \supset A) \supset ((C \supset B) \supset (C \supset (A \wedge B)))$
10. $(A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C))$
11. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$

12. $((A \supset B) \supset A) \supset A$
13. $\neg(A \vee B) \supset (\neg A \wedge \neg B)$
14. $(\neg A \wedge \neg B) \supset \neg(A \vee B)$
15. $\neg(A \wedge B) \supset (\neg A \vee \neg B)$
16. $(\neg A \vee \neg B) \supset \neg(A \wedge B)$
17. $\neg(A \supset B) \supset (A \wedge \neg B)$
18. $(A \wedge \neg B) \supset \neg(A \supset B)$

*The second author is supported by Russian Foundation for Humanities, project 10-03-00570a.

Additionally, a characteristic axiom for PCont is $A \vee \neg A$ while for PComp is $A \wedge \neg A \supset B$.

Rule of inference: From A and $A \supset B$ infer B.

Semantics

The axioms of PCont are adequate to the following matrix semantics: there are three values 1, t, 0 with two designated values 1, t such that $0 < t < 1$ and $A \vee B = max(A, B)$ and $A \wedge B = min(A, B)$. For the $\supset$ and $\neg$ the matrices are given below:

$\supset_{(PCont)}$	1	t	0
1	1	t	0
t	1	t	0
0	1	1	1

p	$\neg_{(PCont)}\ p$
1	0
t	t
0	1

The axioms of PComp are adequate to the following matrix semantics: there are three values $1, f, 0$ with the designated value 1 such that $0 < f < 1$ and $A \vee B = max(A, B)$ and $A \wedge B = min(A, B)$. For the $\supset$ and $\neg$ the matrices are given below:

$\supset_{(PComp)}$	1	f	0
1	1	f	0
f	1	1	1
0	1	1	1

p	$\neg_{(PComp)}\ p$
1	0
f	f
0	1

Below we define the base set of elimination and introduction rules, where prefixes '*el*' and '*in*' denote an elimination and an introduction rule, respectively.

Elimination Rules :

$$\wedge\, el_1 \quad \frac{A \wedge B}{A} \qquad \wedge\, el_2 \quad \frac{A \wedge B}{B}$$

$$\neg\wedge\, el \quad \frac{\neg(A \wedge B)}{\neg A \vee \neg B} \qquad \neg\, el \quad \frac{\neg\neg A}{A}$$

$$\neg\vee\, el_1 \quad \frac{\neg(A \vee B)}{\neg A} \qquad \neg\vee\, el_2 \quad \frac{\neg(A \vee B)}{\neg B}$$

$$\supset el \quad \frac{A \supset B, \quad A}{B} \qquad \neg\supset el_1 \quad \frac{\neg(A \supset B)}{A}$$

$$\neg\supset el_2 \quad \frac{\neg(A \supset B)}{\neg B} \qquad \vee el \quad \frac{A \vee B, [A]C, [B]C,}{C}$$

Introduction Rules :

$$\wedge\, in \quad \frac{A, \quad B}{A \wedge B}$$

$$\neg \wedge\, in_1 \quad \frac{\neg A}{\neg(A \wedge B)} \qquad \neg \wedge\, in_2 \quad \frac{\neg B}{\neg(A \wedge B)}$$

$$\vee\, in_1 \quad \frac{A}{A \vee B} \qquad \vee\, in_2 \quad \frac{B}{A \vee B}$$

$$\neg \vee\, in \quad \frac{\neg A, \neg B}{\neg(A \vee B)} \qquad \supset in \quad \frac{[C] \quad B}{C \supset B}$$

$$\neg \supset in \quad \frac{A, \neg B}{\neg(A \supset B)} \qquad \neg\, in \quad \frac{B}{\neg\neg B}$$

Characteristic Rules

$$PCont \vee\ el \quad \frac{[A]\ C, [\neg A]\ C}{C}$$

$$PComp - \vee\, el \quad \frac{A \vee B,\ \neg A}{B}$$

An inference in these systems is a finite non-empty sequence of formulae such that each formula is an assumption or is derived from the previous ones via a NPCont/NPComp rule; for both logics, by applying $\supset_{in}$ each formula from the last alive assumption until the result of the application of this rule, inclusively, is discarded from the inference; by applying $\vee_{el}$ each formula starting from assumption A until formula C, inclusively, as well as each formula starting from assumption B until formula C, inclusively, is discarded from the inference; by applying $PCont\vee_{el}$ each formula starting from assumption A until formula C, inclusively, as well as each formula starting from assumption $\neg A$ until formula C, inclusively, is discarded from the inference. A proof then is an inference from the empty set of assumptions.

A proof searching algorithm for NPCont

The complete construction of the goal directed proof searching algorithm for NPCont has been presented in [2]. The most important cases here are the following.

$$\Gamma \vdash \Delta, F \longrightarrow \Gamma, \neg F \vdash \Delta, [F]F, [\neg F], F$$

Here we have an unreached goal, F, the last goal in the list of goals, *list_goals*, and the set of formulae in the proof,*list_proof*. This goal F is either a literal or $A \vee B$ or $\neg(A \wedge B)$. When we cannot reach the current goal, F, and other procedures are not applicable, we proceed similar to the classical refutation. However, now, in the setting of paraconsistent logic, we deal with this situation differently. Namely, once we assumed $\neg F$ we aim at achieving the goal F. If this can be done then we can always add to the a proof of F from F. These two inferences would give us the required basis to apply $PCont\vee_{el}$ rule, namely, $[\neg F], F$ and $[F], F$ which would enable us to derive the desired F. Note that although this case invokes an elimination rule, $PCont\vee_{el}$, it has a special role in our heuristic and we consider this rule in line with other introduction rules involved into the searching technique.

Another crucial procedure is invoked when the current goal, F, is not reached and we are looking for the sources of new goals. In this case we search for compound formulae which can serve as sources for new goals. However, unlike in classical case, here we have only two types of compound formulae in *list_proof*: disjunctive and implicative formulae. If one of these formulae is found then the new goal is generated as follows:

$$\Gamma, A \vee B \vdash \Delta, C \longrightarrow \Gamma \vdash \Delta, [A]C \quad \Gamma \vdash \Delta, [B]C$$
$$\Gamma, A \supset B \vdash \Delta, C \longrightarrow \Gamma \vdash \Delta, [A]C \quad \Gamma \vdash \Delta, [\neg A]C$$

Towards a proof searching algorithm for NPComp

Completing the proof search for the paracomplete logic PComp is our recent work. Here we invoke many of the searching procedures developed for classical case and for NPCont. However, the main problem in the new paracomplete setup, where we do not have the law of exluded middle, is to tackle the proof by refutation situation

$$\Gamma \vdash \Delta, F \longrightarrow \Gamma, \neg F \vdash \textbf{false}$$

where since the current goal F is not reachable, we add its negation to *list_proof* targeting to derive a contradiction. We need to find a way to discard this assumption to achieve a completed proof.

References

[1] A. Avron. Natural 3-valued logics–characterization and proof theory. *The Journal of Symbolic Logic*, pages 276–294, March, 1991.

[2] A. Bolotov and V.Shangin. Natural deduction system in paraconsistent setting: Proof search for PCont. *Journal of Intelligent Systems*, 21(1):1–24, February, 2012.

[3] V. Popov. Sequential formulations for paraconsistent logics. *Syntactical and semantic investigations of nonextensional logics*, 1989. (In Russian).

Graphical Construction of Cut Free Sequent Systems Suitable for Backwards Proof Search [*]

Björn Lellmann
bl610@doc.ic.ac.uk

Dirk Pattinson
dirk@doc.ic.ac.uk

Imperial College London, 180 Queen's Gate, London SW7 2AZ

Abstract: We present a graphical representation of sequents and sequent rules, which aids in the discovery of cut-free sequent systems for non-iterative modal logics permitting backwards proof search in polynomial space. The technique is used to construct sequent systems for conditional logic V and KLM rational logic $\mathcal{R}$.

1 Introduction

Backwards proof search is one of the main techniques in theorem proving. The systems used for this usually are sequent systems which have cut elimination and admissibility of contraction. In this context the emergence of ever more specialised modal logics in computer science gives rise to the question of how to construct such systems.

One method of constructing cut-free sequent systems for modal logics is the method of *cut elimination by saturation*, previously used e.g. in [3]. As the name suggests, the method is based on saturating the rule set under the addition of rules which are necessary for the standard cut elimination proof to go through. Unfortunately, in the standard notation for sequent rules the construction of these new rules quickly becomes very tedious and prone to error. In this work we introduce a graphical representation of sequents and sequent rules, which makes the operations needed for saturating a rule set very simple and intuitive. Furthermore, we apply the method to Lewis' conditional logic V from [4] in the entrenchment language, yielding a sequent system suitable for backwards proof search in polynomial space. This sequent system moreover witnesses that deciding the flat fragment of V is in the class Π_3^P of the polynomial hierarchy. By translation this yields a purely syntactical Π_3^P-decision procedure for the KLM rational logic $\mathcal{R}$, which although of suboptimal complexity might still be of interest.

2 Cut Elimination by Saturation and Backwards Proof Search

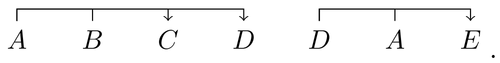
We consider formulae over the propositional connectives, the unary modality $\Box$ and the binary entrenchment modality $\preccurlyeq$. The results easily generalise to other signatures as well. Furthermore we assume the presence of the rules $\mathsf{G3}p$ of [5] for the underlying classical propositional logic. Let's recall some notions and facts from [3]. The general rule format considered is that of a *shallow rule*. Such a rule is given by

$$\frac{\{\Gamma, \Sigma_i \Rightarrow \Delta, \Pi_i \mid i \le n\} \cup \{\Xi_j \Rightarrow \Omega_j \mid j \le m\}}{\Gamma, \Phi \Rightarrow \Delta, \Upsilon},$$

where the $\Sigma_i \Rightarrow \Pi_i$ and $\Xi_j \Rightarrow \Omega_j$ are sequents (i.e. pairs of multisets) of propositional variables, the *contextual* and *noncontextual premisses* respectively, and $\Phi \Rightarrow \Upsilon$ is a sequent of modalised propositional variables, the *principal formulae* of the rule. The sequent $\Gamma \Rightarrow \Delta$ is the *context*. Since axioms without nested modalities can always be converted into equivalent sets of shallow axioms, this format suffices for all logics axiomatised by (finitely many) non-nested axioms. For two rules R_1, R_2 with principal formulae $\Phi_1 \Rightarrow \Upsilon_1, (A \preccurlyeq B)$ and $(A \preccurlyeq B), \Phi_2 \Rightarrow \Upsilon_2$ the rule $\mathsf{cut}(R_1, R_2, (A \preccurlyeq B))$ is the rule with principal formulae $\Phi_1, \Phi_2 \Rightarrow \Upsilon_1, \Upsilon_2$, whose premisses are built by cutting the combined premisses of R_1 and R_2 first on A and then on B. Here exactly those premisses are labelled contextual, whose construction involved at least one contextual premiss of R_1 or R_2. The definition for cuts on formulae of the form $\Box A$ is analogous. It can be shown that the so constructed rules are still sound. A set $\mathcal{R}$ of shallow rules is *cut closed* if for every two rules $R_1, R_2 \in \mathcal{R}$ with principal formulae $\Phi_1 \Rightarrow \Upsilon_1, C$ and $C, \Phi_2 \Rightarrow \Upsilon_2$ the rule $\mathsf{cut}(R_1, R_2, C)$ is derivable in $\mathsf{G3}p\mathcal{R}$ without using the cut rule. Similarly, the rule set is *contraction closed*, if for every rule R with principal formulae $\Phi \Rightarrow \Upsilon, C, C$ there is a rule $R' \in \mathcal{R}$ with principal formulae $\Phi \Rightarrow \Upsilon, C$, whose premisses are derivable from the premisses of R using only contraction and weakening (and similarly for contractions on the left of $\Rightarrow$). The method of cut elimination by saturation is based on the fact that in cut and contraction closed rule sets based on $G3p$ the cut rule can be eliminated, and proof search (in a slightly modified system) can be implemented in polynomial space. Thus the missing cuts between rules and contractions of rules are added until the rule set is saturated.

3 The Graphical Representation

In order to make the process of computing cuts between rules more intuitive we now introduce a graphical representation of sequents. The main idea is to represent a sequent $\Gamma \Rightarrow \Delta$ by a *multiarrow* with *tails* emerging from the formulae in Γ and *heads* pointing to the formulae in Δ. Thus for example the sequents $A, B \Rightarrow C, D$ and $D, A \Rightarrow E$ are represented by the multiarrows



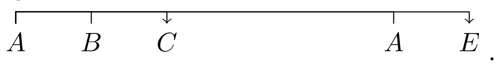
An application of the cut rule to these two sequents with cut formula D now evidently is represented by connecting

[*]Supported by EPSRC-Project EP/H016317/1

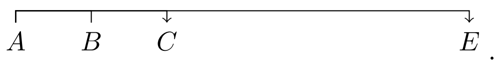
$$\frac{\{\, B_k \Rightarrow A_1,\dots,A_n,D_1,\dots,D_m \mid k \le n \,\} \cup \{\, C_k \Rightarrow A_1,\dots,A_n,D_1,\dots,D_{k-1} \mid k \le m \,\}}{\Gamma,(C_1 \preccurlyeq D_1),\dots,(C_m \preccurlyeq D_m) \Rightarrow \Delta,(A_1 \preccurlyeq B_1),\dots,(A_n \preccurlyeq B_n)}\; R_{n,m}$$

Figure 1: The general rule scheme for the set $\mathcal{R}_V := \{R_{n,m} \mid n \geq 1, m \geq 0\}$.

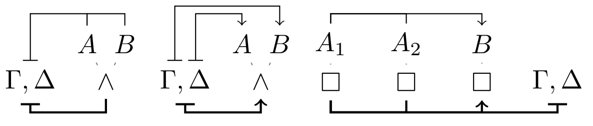
the head of the left multiarrow pointing do D to the tail of the right multiarrow emerging from D, "yanking the wire", and omitting the superfluous instances of the cut formula, resulting in the multiarrow



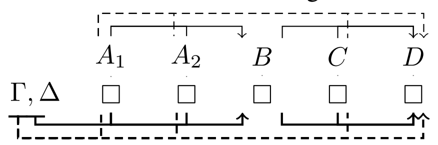
Similarly, the sequent resulting from now contracting the two instances of A is represented by the multiarrow



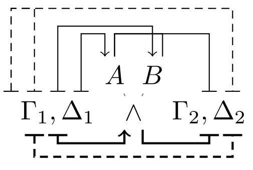
Rules are represented by writing the formulae as parse trees and drawing the multiarrows representing the premisses on top, the multiarrow representing the conclusion on the bottom. To mark contextual premisses we add an additional end to the arrows, marked as ⊣. This is to be read as an abbreviation for tails emerging from the formulae on the left side of the context and heads pointing to those on the right side of the context. Thus for example the left and right conjunction rules of G3p and an instance of the K rule of normal modal logic are represented by



Now the operation of cutting two instances of e.g. the K rule is visualised by performing cuts on the conclusion as well as on the corresponding elements of the premisses resulting in the dashed arrows in the diagram below



If we wanted to saturate the set of K rules, we would now have to add the rule represented by the dashed arrows above to our rule set. On the other hand, if we cut the left and right conjunction rules we get the rule represented by the dashed arrows on the right. Since this is simply an application of contraction, we do not need to add any rules to our rule set. In a similar way we get contractions of rules by contracting their conclusions as well as the corresponding formulae in their premisses.



4 Conditional Logic V and KLM Logic $\mathcal{R}$

This method can be applied to construct a cut- and contraction closed set of sequent rules for the conditional logic V in the entrenchment language. After turning the rules and axioms from [4, p.123,124] into shallow rules using the method mentioned in Section 2 and cutting and contracting rules, we arrive at the set $\mathcal{R}_V$ of sequent rules, whose traditional representation is given in Figure 1. By construction this set is guaranteed to be sound, and since it subsumes the original rules it is also complete. Cut and contraction closure can be seen by considering the graphical representation of the rules. Thus by the results of Section 2 the sequent system $G3p\mathcal{R}_V$ has cut elimination and admissibility of contraction and is suitable for backwards proof search in polynomial space. The translation $(A \mathrel{\Box\!\!\to} B) \equiv (\bot \preccurlyeq A) \vee \neg(A \wedge \neg B \preccurlyeq A \wedge B)$ of the more commonly used counterfactual $\Box\!\!\to$ into the entrenchment connective $\preccurlyeq$ from [4] unfortunately yields a blowup exponential in the nesting depth of $\Box\!\!\to$, but still can be used to show a PSPACE upper bound for the right nested fragment of V in the counterfactual language, thus syntactically reproducing the corresponding result from [2]. Also, since the translation is only linear for formulae without nested modalities, and since the alternation depth in the algorithm for backwards proof search is determined by the nesting depth of the modalities, we get a purely syntactical Π_3^P decision procedure for the flat fragment of V, and by the correspondence from [1] also for KLM rational logic $\mathcal{R}$. Whether this procedure can be pushed down to the optimal complexity coNP is subject of ongoing research.

5 Concluding Remarks

The described method of cut elimination by saturation using the graphical representation has also successfully been used to construct sequent systems for several extensions of V. Furthermore, by incorporating restrictions on the context it is also possible to treat certain nested axioms such as the axioms responsible for absoluteness in V and some examples of intuitionistic modal logics. A more thorough exploration of these issues will be subject of further research.

References

[1] G. Crocco and P. Lamarre. On the connection between non-monotonic inference systems and conditional logics. In *KR*, pp. 565–571, 1992.

[2] N. Friedman and J. Y. Halpern. On the complexity of conditional logics. In *KR*, pp. 202—213, 1994.

[3] B. Lellmann and D. Pattinson. Cut elimination for shallow modal logics. In *TABLEAUX*, pp. 211–225, 2011.

[4] D. Lewis. *Counterfactuals*. Blackwell, 1973.

[5] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 2 edition, 2000.

Automatic Reasoning for Interrogative Logics

Ştefan Minică

stefan.minica@gmail.com

Abstract: We present Interrogative-Epistemic Logic (IEL) and describe an implementation for automatic reasoning inside IEL by means of a tableau prover synthesized from the logic. The paper is structured as follows: In the first part we give a framework that deals with the interdependence between questioning and epistemic aspects in an uniform setting of hybrid logic with nominals and dynamic questioning and resolution modalities reminiscent from [4] and [2]. In the second part we show how a tableau calculus for this logic can be automatically generated using the generic framework introduced in [1] starting from the modal semantics and a background theory for an intersection modality and we describe an implementation in Haskell for the resulting tableau system extending on the previous implementation from [3].

1 Introduction & Motivation

Interrogative or erotetic logics have a long tradition alongside declarative and epistemic logics. While automated reasoning tools are widespread for declarative logics and also for epistemic modal logics, there are almost none designed to deal with interrogative aspects. Since historically logics of questions emerged in their modern form [4, 6] as extended tableau-like calculi tailored to cope with interrogative questioning actions and erotetic inferences, the task of developing an automated reasoning tool for interrogative logic is a natural desideratum. The purpose of the present work is to develop and implement a tableau calculus for logics that can model both interrogative and epistemic aspects in a unified way. Such logics have been studied recently in [2, 5] and they turn out to be variants of hybrid modal logics with nominals and intersection. In the same time, various generic tableau provers [1, 3, 7] for hybrid and intensional logics have been developed lately which provide a general framework in which the specific details emerging in interrogative and questioning contexts of erotetico-epistemic scenarios can be captured. The main contribution in this paper will be to spell out these details and provide an implementation in which automatic reasoning about them can be performed. This will provide a tool that can be useful in many applications ranging from planning for epistemic-erotetic agents to designing efficient querying strategies and optimal inquiry procedures in science, and query-driven rational interaction in general.

2 Interrogative Epistemic Logic

The language of Interrogative-Epistemic Logic (henceforth, IEL) is recursively defined by the following BNF:

$$\varphi \quad ::= \quad n \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid [Q]\varphi$$

with $\Box \in \{Q_a, X_a, K_a\}$ static modalities, $Q \in \{\varphi?, !\}$ representing dynamic actions and n, p, a standing for nominals, propositional symbols and agent labels, respectively. The language can express the interaction between questions and information in two ways. First by using a (static) intersection modality $X_a\varphi$. Second through the dynamic modalities $[Q]$ encoding model-changing operations by means of questioning and resolution actions. This basic language can be extended in many ways when needed. The most common additions include the *set* operator $\{n\}$, the *at* (or *satisfaction*) operator $@_n\varphi$, the slashed disjunction $\vee/\diamond$, various modalities for group notions of knowledge, questioning or both, the down-arrow binder $\downarrow x.\varphi$, etc.

The language has a standard modal semantics over issue-epistemic structures $M = \langle W, \approx, \sim, V\rangle$ using the expected Boolean clauses and the usual relational (modal) clauses with $\approx$ for Q and $\sim$ for K. The intersection modality X is defined using $\approx\cap\sim$ in the following way:

$$M \models_w X_a\varphi \quad \text{iff} \quad \forall v \in W : w\,(\overset{a}{\approx}\cap\overset{a}{\sim})\,v \Rightarrow M \models_v \varphi$$

The dynamic modalities $[\varphi?]\psi$ and $[!]\varphi$ express model-changing operations resulting in new models $M_?$ and $M_!$ in which $\approx_? = \approx \cap \overset{\varphi}{\equiv}_M$ and $\sim_! = \sim \cap \approx$, respectively, while all the other components remain unchanged. The intuitive reading for the questioning modality $[\varphi?]\psi$ is "after φ is asked, ψ is the case". The intuitive reading for the resolution modality $[!]\varphi$ is "after all the questions raised so far have been answered, φ is holds". Dynamic modalities of announcement $[\varphi!]\psi$ and refinement $[?]\varphi$ are sometimes also considered in this setting, their behaviour is completely symmetric, changing $\sim$ respectively $\approx$ correspondingly.

The static fragment of the logic is axiomatised by a customary hybrid logic system with nominals, S5 axioms for $\sim$ and $\approx$, and an intersection axiom for static resolution:

$$\widehat{K}i \wedge \widehat{Q}i \leftrightarrow \widehat{R}i, \qquad \text{where } i \text{ is a nominal}$$

Formulas containing dynamic modalities can be reduced to equivalent static formulas using reduction axioms like:

$$[\varphi?]Q\psi \leftrightarrow (\varphi\wedge Q(\varphi \to [\varphi?]\psi))\vee(\neg\varphi\wedge Q(\neg\varphi \to [\varphi?]\psi))$$

$$[\varphi?]X\psi \leftrightarrow (\varphi\wedge X(\varphi \to [\varphi?]\psi))\vee(\neg\varphi\wedge X(\neg\varphi \to [\varphi?]\psi))$$

$$[\varphi?]K\psi \leftrightarrow K[\varphi?]\psi,\ \ [!]K\varphi \leftrightarrow X[!]\varphi,\ \ [!]Q\varphi \leftrightarrow Q[!]\varphi$$

We refer to [2, 5] for further technical details, possible extensions and examples of applications for the logic.

$$(a)\ \frac{\nu_1(\bot, x)}{\bot},\ \frac{\neg\nu_1(\bot, x)}{\neg\bot};\quad \frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)},\ \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)};\quad \frac{\nu_1(p \vee q, x)}{\nu_1(p, x) \mid \nu_1(q, x)},\ \frac{\neg\nu_1(p \vee q, x)}{\neg\nu_1(p, x)\,,\ \neg\nu_1(q, x)};$$

$$(b)\ \frac{\nu_1(Qp, x)}{\neg R_{\approx}(x, y) \mid \nu_1(p, y)};\ \frac{\neg\nu_1(Qp, x)}{R_{\approx}(x, f(p, x))\,,\ \neg\nu_1(p, f(p, x))};\quad (c)\ \frac{R_{\approx\cap\sim}(x, y)}{R_{\approx}(x, y), R_{\sim}(x, y)},\ \frac{R_{\approx}(x, y), R_{\sim}(x, y)}{R_{\approx\cap\sim}(x, y)};$$

$$(d)\ \frac{x = x}{R(x, x)},\ \frac{R(x, y)}{R(y, x)},\ \frac{R(x, y), R(y, z)}{R(x, z)};\quad (e)\ \frac{}{x = x},\ \frac{x = y}{y = x},\ \frac{x = y,\ y = z}{x = z};\quad (f)\ \frac{i = j}{\nu_0(i) = \nu_0(j)};$$

$$(g)\ \frac{x = y,\ \nu_1(p, x)}{\nu_1(p, y)},\ \frac{x = y}{\nu_1(p, f(x)) = \nu_1(p, f(y))};\ (h)\ \frac{\nu_1(p, x)\,,\ \neg\nu_1(p, x)}{\bot},\ \frac{R_{\approx}(x, y)\,,\ \neg R_{\approx}(x, y)}{\bot};$$

$$(i)\ \frac{\nu_1([\varphi?]Q\psi, x)}{\varphi \wedge Q(\varphi \to [\varphi?]\psi) \mid \neg\varphi \wedge Q(\neg\varphi \to [\varphi?]\psi)},\ \frac{\nu_1([\,!\,]K\varphi, x)}{\nu_1(X[\,!\,]\varphi, x)},\ \frac{\nu_1(\neg[\,!\,]K\varphi, x)}{\nu_1(\neg X[\,!\,]\varphi, x)},\ \frac{\nu_1([\varphi?]a, x)}{\nu_1(a, x)},\ \frac{\nu_1(\neg[\varphi?]a, x)}{\nu_1(\neg a, x)}.$$

Figure 1: Tableau Rules for Interrogative-Epistemic Logic Synthesized from the Modal Semantics & Background Theory

3 Implementing a Tableau Prover for IEL

In this section we synthesize the tableau rules for IEL and briefly describe the resulting implementation using these rules. The general method we followed is the one introduced in [1] and consists of the following steps. Defining an object-language and a metalanguage with specific sorts for propositions, nominals, and the domain sort based on which the semantics of the modal connectives can be specified. Spelling out the definitions of the specific logical connectives and the particular background theory. Finally, generating and refining tableau rules from the normalised specifications. After the formal details specific for IEL are established the generic framework provides automatically desirable metaproperties for the resulting tableau calculus.

The resulting rules are presented in Figure 1 and they include the expected decomposition or expansion rules, synthesized from the semantic definitions of the Boolean (a) connectives. Analogously for the questioning modal logical connectives (b). Theory rules synthesised from the background theory containing the crucial intersection axiom (c). The usual axioms for epistemic and issue equivalence relations (d). The usual rules for equality and substitutivity $(e$-$g)$. Standard closure rules (h). Rules for dynamic modalities synthesized from the reduction axioms. For the sake of brevity the $\nu_1(\cdot, x)$ pattern was omitted in both denominator formulae at (i). The rules in (i) merely push the dynamic modalities inside until the rules of the static fragment will eventually perform the final logical decomposition.

The final step consists in translating theoretical aspects in implementation details. We only briefly describe these here. We started from the preexisting tableau prover for hybrid logic developed in [3]. Here frame conditions are imposed by adding formulae with universal modality to the set of starting sets of formulae. We replaced this by adding intersection rules for the background theory, we also switched from equality constraints to unrestricted blocking, which are both more congenial with the setting from [1].

4 Conclusions and Further Work

The paper introduces a logic that describes interrogative and epistemic aspects in an uniform setting using hybrid logic with nominals and intersection. A tableau calculus for this logic was generated using the generic framework from [1] and an implementation for it was developed.

Topics on our future research agenda include the following. Adding a richer repertoire of questioning actions. A study of connections between search heuristic and principles for designing efficient query strategies. Implementing the resulting rules as an extension module inside the MetTel architecture. Using the resulting implementation as a framework for studying strategic aspects in questioning procedures and connections with notions of relevance for questions and epistemic entropy. Investigating illustrative applications of the resulting tableau system.

References

[1] R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Log. Met. in Co. Sc.*,7(2):1–32, 2011.

[2] Ş. Minică. *Dynamic Epistemic Logic of Questions in Inquiry*. Ph.D. Thesis, Univ. of Amsterdam, 2011.

[3] J. van Eijck. Hylotab – tableau-based theorem proving for hybrid logics. *Manuscript, CWI, Amsterdam*, 2002.

[4] J. Hintikka, I. Halonen, and A. Mutanen. Interrogative logic as a general theory of reasoning. *St. in Logic and Practical Reasoning*, 1:295–337, 2002.

[5] J. van Benthem and Ş. Minică. Toward a Dynamic Logic of Questions. *J. of Phil. Logic*, 2011.

[6] A. Wiśniewski. Erotetic search scenarios. *Synthese*, 134(3):389–427, 2003.

[7] L. del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, F. Massacci. Lotrec: a generic tableau prover for modal and description logics. *Aut. Reas.*, p. 453, 2001.

Terminating tableau calculus for the logic $\mathbf{K}(\mathsf{E}_n)$

Michał Zawidzki*

[1] Institute of Philosophy, University of Lodz
[2] School of Computer Science, The University of Manchester
`michal.zawidzki@gmail.com`

Abstract: Logic $\mathbf{K}(\mathsf{E}_n)$ is a powerful extension of the ordinary modal logic **K**. Due to adding global counting modalities, expressive power of the logic covers all most popular hybrid logics ($\mathcal{H}(@)$, $\mathcal{H}(\mathsf{E})$, $\mathcal{H}(\mathsf{D})$), as well as the graded modal logic $\mathbf{K}_n$. In this abstract a refined version of the tableau calculus for $\mathbf{K}(\mathsf{E}_n)$ is presented, in which we encode semantic expressions (like *labels*) within the $\mathbf{K}(\mathsf{E}_n)$-language.

1 Introduction

Modal logics with counting operators were first introduced by Fine ([3]) under the name of "graded modal logics". The idea standing behind introducing graded modalities was to enable counting the number of successors of a particular world (which was not achievable by means of ordinary modal logic). Nevertheless, it only allowed to operate within a "local" range (along the edges of accessibility relation). Even though the modal logic **S5** with graded modalities could be looked upon as making possible to count a number of worlds in the whole model, it lacked the ordinary modalities instead. That was the reason of introducing global counting modalities apart from ordinary ones. Areces et al. ([1]) named this calculus "modal logic with counting". They also provided a program that should decide whether a given formula has a model. Unfortunately, it exploited the translation function from the logic $\mathbf{K}(\mathsf{E}_n)$ (modal logic **K** with counting) to $\mathcal{H}(\mathsf{E})$ (hybrid logic with counting modality), which leads to exponential blow-up if numbers are coded in binary.

Calculi with counting operators were widely investigated in the field of description logics where counting modalities were called *cardinality restrictions*. There are several interesting tableau systems for description logics with cardinalities ($\mathcal{ALCQ}$, $\mathcal{SHCQ}$), see e.g. [4], [2], but either they exploit the tools that are not available in $\mathbf{K}(\mathsf{E}_n)$ or blocking mechanisms used in the calculi are conceptually complex.

In this abstract we outline the terminating tableau calculus for $\mathbf{K}(\mathsf{E}_n)$ which is complexity-optimal and utilises the *unrestricted blocking* mechanism (introduced by Schmidt and Tishkovsky in [5] and [6]) which is both conceptually simple and effective.

We present the refined version of the calculus with no prefixes or labels. The only expressions that occur in the rules of the calculus are $\mathbf{K}(\mathsf{E}_n)$-expressions.

2 Logic $\mathbf{K}(\mathsf{E}_n)$

Let $\text{PROP} = \{p_1, p_2, \ldots\}$ be a countable set of propositional letters. We define a set FORM of formulas of $\mathbf{K}(\mathsf{E}_n)$ as follows:

$$\text{FORM} ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi \mid \mathsf{E}_{>n}\varphi, \qquad (\mathbf{K}(\mathsf{E}_n))$$

where $p \in \text{PROP}$, $\varphi \in \text{FORM}$, $n \in \mathbb{N}$. Logic $\mathbf{K}(\mathsf{E}_n)$ does not allow transfinite numbers in subscripts of $\mathsf{E}_{>n}$ operators.

Other counting connectives are defined as follows:

$$\mathsf{E}_{<n} := \neg\mathsf{E}_{>n-1} \qquad \mathsf{E}_{=n} := \mathsf{E}_{>n-1} \wedge \neg\mathsf{E}_{>n}\varphi$$

A model for $\mathbf{K}(\mathsf{E}_n)$ is a triple $\langle W, R, V\rangle$ where W is a non-empty set, R is a binary relation on W, $V : \text{PROP} \to \mathcal{P}(W)$ is a valuation function assigning to each $p \in \text{PROP}$ a set of worlds $w \in W$ in which p holds. Given a model $\langle W, R, V\rangle$ and $w \in W$, the semantics for $\mathbf{K}(\mathsf{E}_n)$ is defined in Figure 1.

$$\begin{array}{lll}
\mathfrak{M}, w \models p & \text{iff} & w \in V(p),\ p \in \text{PROP} \\
\mathfrak{M}, w \models \neg\varphi & \text{iff} & \mathfrak{M}, w \not\models \varphi \\
\mathfrak{M}, w \models \varphi \wedge \psi & \text{iff} & \mathfrak{M}, w \models \varphi \text{ and} \\
\mathfrak{M}, w \models \Diamond\varphi & \text{iff} & \text{there is a } v \text{ s. t. } wRv \text{ and} \\
 & & \mathfrak{M}, v \models \varphi \\
\mathfrak{M}, w \models \mathsf{E}_{>n}\varphi & \text{iff} & \|\{w : \mathfrak{M}, w \models \varphi\}\| > n,
\end{array} \qquad (1)$$

where $\|A\|$ means the cardinality of a set A.

3 Tableau calculus $\mathcal{T}^{R_d}_{\mathbf{K}(\mathsf{E})_n}$

As it occurs, expressive power of the language of $\mathbf{K}(\mathsf{E}_n)$ is sufficient to express semantics of the logic within. Due to the fact that $\mathsf{E}_{>n}$-operators combine counting properties and global range, we are able to bypass explicit labelling expressions by exploiting $\mathsf{E}_{=1}$-operator in an appropriate way. Thus, for each input set of formulas Γ we can encode semantic expressions as follows:

$$\begin{array}{l}
\mathbf{K}(\mathsf{E}_n)(\mathfrak{M}, x \models \varphi) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=0}(p_x \wedge \neg\varphi)^* \\
\mathbf{K}(\mathsf{E}_n)(\mathfrak{M}, x \not\models \varphi) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=0}(p_x \wedge \varphi)^* \\
\mathbf{K}(\mathsf{E}_n)(xRy) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=1}p_y \wedge \mathsf{E}_{=0}(p_x \wedge \neg\Diamond p_y)^* \\
\mathbf{K}(\mathsf{E}_n)(\neg xRy) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=1}p_y \wedge \mathsf{E}_{=0}(p_x \wedge \Diamond p_y)^* \\
\mathbf{K}(\mathsf{E}_n)(x = y) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=1}p_y \wedge \mathsf{E}_{=0}(p_x \wedge \neg p_y)^* \\
\mathbf{K}(\mathsf{E}_n)(x \neq y) = \mathsf{E}_{=1}p_x \wedge \mathsf{E}_{=1}p_y \wedge \mathsf{E}_{=0}(p_x \wedge p_y)^*
\end{array} \qquad (2)$$

$^*p_i, p_j$ are fresh prop. variables not occurring in Γ.

*The research reported in this abstract is a part of the project financed from the funds supplied by the **National Science Centre, Poland** (decision no. **DEC-2011/01/N/HS1/01979**)

Rules for the connectives:

$$(\neg\neg)\ \frac{\varphi : \neg\neg\psi}{\varphi : \psi} \quad (\wedge)\ \frac{\varphi : \psi \wedge \chi}{\varphi : \psi, \varphi : \chi} \quad (\neg\wedge)\ \frac{\varphi : \neg(\psi \wedge \chi)}{\varphi : \neg\psi \mid \varphi : \neg\chi} \quad (\Diamond)\ \frac{\varphi : \Diamond\psi}{\varphi : \Diamond f(\Diamond\psi, \varphi), f(\Diamond\psi, \varphi) : \psi} \quad (\neg\Diamond)\ \frac{\varphi : \neg\Diamond\psi, \varphi : \Diamond\chi, \chi : \chi}{\chi : \neg\psi}$$

$$(\mathsf{E}_{>n})\ \frac{\varphi : \mathsf{E}_{>n}\psi}{f_1(\mathsf{E}_{>n}\psi, \varphi) : \psi, \dots, f_{n+1}(\mathsf{E}_{>n}\psi, \varphi) : \psi \underset{0<k<l\leq n+1}{,} f_k(\mathsf{E}_{>n}\psi, \varphi) : \neg f_l(\mathsf{E}_{>n}\psi, \varphi)} \quad (\neg\mathsf{E}_{>n})\ \frac{\varphi : \neg\mathsf{E}_{>n}\psi, \chi_1 : \chi_1, \dots, \chi_{n+1} : \chi_{n+1}}{\chi_1 : \neg\psi \mid \dots \mid \chi_{n+1} : \neg\psi \underset{0<k<l\leq n+1}{\mid} \chi_k : \neg\chi_l}$$

Rules for equality, closure rules, unrestricted blocking rule:

$$\frac{\varphi : \psi}{\varphi : \varphi} \quad \frac{\varphi : \psi, \psi : \psi}{\psi : \varphi} \quad \frac{\varphi : \psi, \psi : \psi, \chi[\varphi]}{\chi[\psi/\varphi]} \quad \frac{\varphi : \psi, \psi : \psi}{f(\chi, \varphi) : f(\chi, \psi)} \qquad \frac{\varphi : \bot}{\bot} \quad \frac{\varphi : \psi, \varphi : \neg\psi}{\bot} \qquad (\text{ub})\ \frac{\varphi : \varphi, \psi : \psi}{\varphi : \psi \mid \varphi : \neg\psi}$$

Figure 1: Rules for refined calculus $\mathcal{T}^{R_d}_{\mathbf{K}(\mathsf{E})_n}$

The only thing that we need to show is that the forgoing expressions actually encode the domain sort expressions.

Proposition 1. *Semantic expressions $\mathfrak{M}, x \models \varphi$, xRy, $x = y$ hold for a model $\mathfrak{M}$ if, and only if respective $\mathbf{K}(\mathsf{E}_n)$-expressions from (2) are satisfiable on a suitable conservative extension $\mathfrak{M}'$ of $\mathfrak{M}$.*

Remark 1. Henceforth, we introduce new *colon* notation. We abbreviate formulas of the form: $\mathsf{E}_{=1}\varphi \wedge \mathsf{E}_{=0}(\varphi \wedge \neg\psi)$ to $\varphi : \psi$.

Now we define a countable set of the form: $\mathcal{F} = \{f_j\}_{j\in\mathbb{N}}$ where $f_j : \text{FORM} \times \text{FORM} \mapsto \text{FORM}$ is a function. We therefore introduce a countable set of functional symbols to obtain an expression class analogous to a class of Skolem functions in first-order language.

Figure 1 presents rules for the refined calculus $\mathcal{T}^{R_d}_{\mathbf{K}(\mathsf{E})_n}$. Profit that is yielded by expressing semantics by means of the logic is a low number of rules.

The *colon* notation introduced in the forgoing refined tableau calculus resembles standard prefixed calculi. For a given input set of formulas Γ a label of the initial node is a propositional variable obtained by translating $\mathfrak{M}, x \models \bigwedge\Gamma$ in a way from (2). New labels introduced by the rules $(\Diamond)$ and $(\mathsf{E}_{>n})$ are arbitrary formulas (by definition of functional symbols f_i). The reader might be surprised by the rule $(\Diamond)$, since one of formulas in the conclusion is of the same form as a premiss-formula, which can lead to an infinite derivation. What distinguishes these two is the fact that a formula which is under the scope of $\Diamond$ in the premiss is not necessarily a labelling formula, whereas a parallel formula in the conclusion certainly is. This makes it subjected to application of (ub) and equality rules which ensures finiteness (with respect to $(\Diamond)$-application) of at least one of the branches.

Theorem 1. $\mathcal{T}^{R_d}_{\mathbf{K}(\mathsf{E})_n}$ *is sound and complete.*

4 Termination and complexity-optimality

Lemma 1. *Let Γ be an arbitrary set of $\mathbf{K}(\mathsf{E})_n$-formulas. Suppose that $\mathfrak{N} = \langle \mathbf{U}, \mathbf{S}, \mathbf{z} \rangle$ is a model for Γ. Then there exists a branch $\mathcal{B}$ in $\mathcal{T}^{R_d}_{\mathbf{K}(\mathsf{E})_n}$-tableau for Γ such that $\|\mathbf{U}\| \geq \|\mathcal{B}\|$.*

Theorem 2. *Logic* $\mathbf{K}(\mathsf{E}_n)$ *has the* effective finite model property *with the bounding function $\mu = 2^{\|\{\text{Sub}(\varphi)\}\| + \log(n+1)}$ for any given input formula φ, where* $\text{Sub}(\varphi)$ *is a set of all subformulas of φ and $n = \max\{m : \mathsf{E}_{>m} \in \text{Sub}(\varphi)\}$, n coded in binary.*

Theorem 3. $\mathcal{T}^{(ub)}_{\mathbf{K}(\mathsf{E})_n}$ *is terminating.*

Theorem 4. $\mathbf{K}(\mathsf{E}_n)$ *is* NEXPTIME*-complete.*

To provide a complexity-optimal derivation strategy for $\mathcal{T}^{(ub)}_{\mathbf{K}(\mathsf{E})_n}$ we formulate the following condition:

(op) Expand a branch of $\mathcal{T}^{(ub)}_{\mathbf{K}(\mathsf{E})_n}$-tableau until the size of $\mathcal{B}$ exceeds the bound from theorem 2. Then stop.

Acknowledgements

This abstract is a result of a series of seminars with dr Renate Schmidt, dr Dmitry Tishkovsky and Mohammad Khodadadi.

References

[1] C. Areces, G. Hoffmann, and A. Denis. Modal Logics with Counting. In *Proc. of WoLLIC 2010*, Brasilia, Brazil, 2010.

[2] J. Faddoul, N. Farsinia, V. Haarslev, and R. Möller. A hybrid tableau algorithm for $\mathcal{ALCQ}$. In *Proc. of ECAI 2008*, Amsterdam, The Netherlands, 2008.

[3] K. Fine. In so many possible worlds. *Notre Dame J. of Form. Log.*, 13(4), 1972.

[4] M. Kaminski, S. Schneider, and G. Smolka. Terminating tableaux for graded hybrid logic with global modalities and role hierarchies. *Log. Meth. in Comp. Sc.*, 7(1:5), 2011.

[5] R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. volume 4825, Busan, Korea, 2007.

[6] R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Log. Meth. in Comp. Sc.*, 7(2), 2011.

A Labelled Tableau Approach for Temporal Logic with Constraints

Clare Dixon[1] Boris Konev[1] Renate A. Schmidt[2] Dmitry Tishkovsky[2]*

[1] Department of Computer Science, University of Liverpool, Liverpool, UK
{CLDixon, Konev}@liverpool.ac.uk
[2] School of Computer Science, University of Manchester, Manchester, UK
{Dmitry.Tishkovsky, Renate.Schmidt}@manchester.ac.uk

Abstract: Frequently when formalising dynamic systems, we must represent statements, coming from physical constraints or representational issues, stating that exactly n literals (or less than n literals) of a set hold. While we can write temporal formulae to represent this information, such formulae both complicate and increase the size of the specification and adversely affect the performance of provers. In this paper, we consider reasoning about problems specified in propositional linear time temporal logics in the presence of such constraints on literals. We present a sound, complete and terminating tableau calculus which embeds constraints into its construction avoiding their explicit evaluation. We use METTEL^2, an automated tableau prover generator, to provide an implementation of the calculus and give experimental results using the prover. This paper is an abstract of the results and methods of [3].

1 Introduction

Temporal logics have been used to represent and reason about systems that change over time [1, 4]. Often when representing such systems we need to formalise that exactly n or less than or equal to m propositions from a set hold or do not hold. This may come about for two reasons. First, they may represent real constraints in the world, for example n machines are available to perform a task or a room has capacity of at most m. Alternatively, they may come about from representational issues. For example, consider a number of robots moving about a grid. Each robot may occupy exactly one square in the grid, so, if $at(i)_{x,y}$ denotes that the robot i is in square (x, y), then exactly one of the propositions $at(i)_{0,0}, at(i)_{0,1}, \ldots at(i)_{n,n}$ will hold, for each robot, for an $n \times n$ grid.

One way to deal with such constraints is to rewrite them as temporal formulae, for example, if we are required to make exactly one from the set $\{p, q, r\}$ true we can represent this as the formulae $\Box(p \lor q \lor r)$, $\Box(\neg p \lor \neg q)$, $\Box(\neg p \lor \neg r)$, $\Box(\neg q \lor \neg r)$, where $\Box$ is the operator 'at every moment in time' from propositional linear-time temporal logic (PTL). However, introducing such additional formulae lengthen and complicate the specification and adversely affect the performance of provers. Instead, we consider a logic, called TLC, introduced in [2]. TLC is a propositional linear-time temporal logic that allows sets of constraints as input.

The aim of this paper is twofold. First, it builds on and extends the work in [2] and its related implementation in a number of ways. We develop a tableau calculus that includes rules for reasoning about constraints in a more goal-directed, incremental way. This retains the advantages of using constraints and overcomes the disadvantages encountered in the approach from [2]. In particular, the new calculus does not require its input to be in a particular normal form so additional unconstrained propositions are not introduced. We are not always forced to explicitly enumerate the constraints. Further we can construct the tableau derivation branch by branch in a depth-first left-to-right manner.

Second, we are interested in exploring the possibilities of extending our METTEL^2 tableau prover generation technology [8] to temporal logic. METTEL^2 presents a first step to implement the tableau calculus synthesis framework introduced in [6] and extends it to a tableau prover generation platform. Temporal logic is an interesting case study for the tableau synthesis endeavour because logics with eventualities such as temporal logics cannot be handled with standard, essentially first-order tableau approaches. In the current form the tableau calculus synthesis framework is not applicable to temporal logic or other logics with eventualities. In this paper we show however how the tableau prover generator METTEL^2 can be used to generate an implemented prover for the tableau calculus introduced for TLC. Here we use the standard representation of the semantics for PTL which involves fix-point operators.

2 Cardinality constraints and tableau calculus

A cardinality constraint $C^{\propto m}$ is a tuple $(C, \propto, m)$, where C is a set of literals (propositions and their negations), $\propto \in \{=, \leq\}$ and $m \in \mathbb{Z}$. The size $\#C$ of a literal set C is the number of literals in C.

A constraint $C^{\leq m}$ (resp. $C^{=m}$)is *valid* in a PTL-model if for every moment of time of the model, there are less than or equal to m (resp. exactly m) literals in C which are true at the given time moment. A constraint set $\mathcal{C}$ is *valid* in a PTL-model if every constraint from $\mathcal{C}$ is valid in the model. A PTL-formula ϕ is satisfiable with respect to a set of constraints $\mathcal{C}$ if it is satisfiable in a PTL-model which validates $\mathcal{C}$.

In order to solve the problem of satisfiability of PTL-formulae with respect to a constraint set we introduced a

*Corresponding author.

tableau calculus (see [3] for details). The tableau calculus T_{TLC} for TLC is based on known tableau calculi for PTL (see e.g. [7]). The calculus has two distinguished features. Namely, it uses a special technique for handling eventualities of fix-point operators and special rules for dealing with boolean constraints.

Our technique to handle eventualities is inspired by algorithms which are commonly used for calculating fix-points in the μ-calculus [5]. Bad loops and good loops in the tableau derivation are identified by checking that all eventualities in the current branch are fulfilled. We extend the logical language with connectives, $E_\Diamond$ and $E_{\mathcal{U}}$ which are introduced via the following rules.

$$(\Diamond){:}\frac{@_\ell \Diamond\phi}{@_\ell E_\Diamond(\phi)} \quad (\mathcal{U}){:}\frac{@_\ell \phi\,\mathcal{U}\,\psi}{@_\ell E_{\mathcal{U}}(\phi,\psi)}$$

Special rules $(E_\Diamond)$ and $(E_{\mathcal{U}})$ for unravelling the eventualities are as follows.

$$(E_\Diamond){:}\frac{@_\ell E_\Diamond(\phi)}{@_\ell\phi,\ E_\Diamond(\phi)\approx\Diamond\phi \mid @_{f(\ell)}E_\Diamond(\phi)}$$
$$(E_{\mathcal{U}}){:}\frac{@_\ell E_{\mathcal{U}}(\phi,\psi)}{@_\ell\psi,\ E_{\mathcal{U}}(\phi,\psi)\approx(\phi\,\mathcal{U}\,\psi) \mid @_\ell\phi,\ @_{f(\ell)}E_{\mathcal{U}}(\phi,\psi)}$$

It is important that the left conclusions of these rules contain equality expressions and since METTEL^2 supports rewriting of arbitrary (ground) expressions the rules trigger rewriting in the left derived nodes. In this case, the particular eventuality expression $E_\Diamond(\phi)$, respectively $E_{\mathcal{U}}(\phi,\psi)$, is rewritten and disappears from the node indicating that eventuality $\Diamond\phi$, respectively $\phi\,\mathcal{U}\,\psi$, is fulfilled. In the right derived node the rules $(E_\Diamond)$ and $(E_{\mathcal{U}})$ leave the expressions $E_\Diamond(\phi)$, respectively, $E_{\mathcal{U}}(\phi,\psi)$, untouched indicating that the eventualities are not fulfilled yet. The other introduced rules ($E_\Diamond$-test) and ($E_{\mathcal{U}}$-test)

$$(E_\Diamond\text{-test}){:}\frac{@_\ell E_\Diamond(\phi)}{\bot} \qquad (E_{\mathcal{U}}\text{-test}){:}\frac{@_\ell E_{\mathcal{U}}(\phi,\psi)}{\bot}$$

are applied only when no other rules are applicable. These rules close the branch in the case that some eventuality is still not fulfilled in the branch. The intuition of this technique can be seen from the representation of the fix-point operator in the μ-calculus and, thus, it can be extended to other logics with fix-point operators, for example, propositional dynamic logic and the full μ-calculus.

The rules for boolean constraints are the following.

$$(C^+_-){:}\frac{@_\ell(\{p\}\cup C)^{\propto m},\ @_\ell\neg p}{@_\ell C^{\propto m}} \quad (C^+_+){:}\frac{@_\ell(\{p\}\cup C)^{\propto m},\ @_\ell p}{@_\ell C^{\propto (m-1)}}$$
$$(C^-_+){:}\frac{@_\ell(\{\neg p\}\cup C)^{\propto m},\ @_\ell p}{@_\ell C^{\propto m}} \quad (C^-_-){:}\frac{@_\ell(\{\neg p\}\cup C)^{\propto m},\ @_\ell\neg p}{@_\ell C^{\propto (m-1)}}$$
$$(\text{cut}^+){:}\frac{@_\ell(\{p\}\cup C)^{\propto m}}{@_\ell p \mid @_\ell\neg p} \quad (\text{cut}^-){:}\frac{@_\ell(\{\neg p\}\cup C)^{\propto m}}{@_\ell p \mid @_\ell\neg p}$$
$$(\text{empty}){:}\frac{@_\ell C^{\propto -1}}{\bot} \quad (\text{cap}){:}\frac{@_\ell C^{=m},\ \#C<m}{\bot}$$

To understand these rules suppose that $\propto$ denotes $=$. The intuition of the rule (C^+_-) is as follows: if $(\{p\}\cup C)^{=m}$ is true and $\neg p$ is true, that is, p is not true, then $C^{=m}$ must be true. The rule (C^+_+) says that if $(\{p\}\cup C)^{=m}$ and p are true then $C^{=(m-1)}$ must be true. The rules (C^-_+) and (C^-_-) are duals. The intuitions are similar for $\propto$ denoting $\leq$. The (cut$^+$) and (cut$^-$)-rules are DPLL type analytic cut rules enumerating the possible truth assignments to propositional symbols occurring in constraints. The rule (empty) closes a branch for a constraint $C^{=-1}$ or $C^{\leq -1}$, because constraints cannot contain a negative number of literals. The (cap)-rule is an early closure rule for the case that m literals of a constraint must hold but the constraint already contains less than m literals.

It is proved in the extended version of this paper [3] that T_{TLC} is sound, complete and terminating. The termination is achieved via the unrestricted blocking mechanism [6].

Theorem 1 *T_{TLC} is sound, complete and terminating tableau calculus for the logic* TLC*.*

3 Concluding remarks

The LTLC prover used in the experiments of this paper can be obtained from the METTEL^2 Demo webpage at `www.mettel-prover.org`. In a web-interface provided it is possible to select both PTL and TLC as predefined logics with predefined tableau calculi for which then a tableau prover can be generated at the click of a button. The user can then download the generated prover or use it directly via the web-interface.

References

[1] H. Barringer, M. Fisher, D. Gabbay, and G. Gough, editors. *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*. Kluwer, 2000.

[2] C. Dixon, M. Fisher, and B. Konev. Temporal logic with capacity constraints. In *FroCoS'07*, volume 4720 of *LNCS*, pages 163–177. Springer, 2007.

[3] C. Dixon, B. Konev, R. A. Schmidt, and D. Tishkovsky. A labelled tableau approach for temporal logic with constraints. Available at `www.mettel-prover.org/papers/dkst12.pdf`.

[4] M. Fisher, D. Gabbay, and L. Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[5] D. Kozen. Results on propositional μ-calculus. *Theoret. Computer Sci.*, 27(3):333–354, 1983.

[6] R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Logical Methods in Comput. Sci.*, 7(2):1–32, 2011.

[7] S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Tableaux'98*, volume 1397 of *LNAI*, pages 277–291. Springer, 1998.

[8] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. MetTeL2: Towards a tableau prover generation platform. Available at `http://www.mettel-prover.org/papers/MetTeL2SysDesc.pdf`.

Minimal Models for Modal Logics

Fabio Papacchini Renate A. Schmidt

School of Computer Science, The University of Manchester {papacchf,schmidt}@cs.man.ac.uk

Abstract: Much research into automated generation of models and minimal models has been carried out. For historical reasons, minimal model generation for classical logics has received most of the attention, but the increasing interest and use of non-classical logics in Computer Science bring the necessity to generalise results and techniques for classical logics to non-classical logics. In this short abstract we present and discuss possible minimality criteria for modal logics.

1 Introduction

For classical logics, propositional logic and first-order logic, several studies into minimal model generation have already been performed (e.g. [2, 5, 7]). It is noticeable clear that there is no unique definition of the notion of a minimal model. A possible categorisation of minimality criteria is the following: domain minimality (e.g. [5]); generation of minimal Herbrand models (e.g. [2]); generation of models that are minimal with respect to a certain set of predicates (e.g. [6]).

As many non-classical logics are translatable into fragments of first-order logic, it is easy to think of a way to apply the same minimality criteria also to such non-classical logics by using a translation-based approach ([3]). Nevertheless, just relying on a translation approach may be too restrictive: how should minimality criteria for classical logics be adapted for non-classical logics? Are there other meaningful minimality criteria for non-classical logics?

In this short abstract we discuss what is the meaning of the minimality criteria used in classical logics from a modal logic point of view, and we present a brief discussion of another possible minimality criterion that is more specific for modal logics (namely, minimality with respect to bisimulation).

While presenting different minimality criteria we refer to Figure 1 to have a visual idea of the generated models and the differences between them. It is worth to say that the models shown in figure are not all the possible models of the formula under consideration, but they suffice for our purposes.

2 Minimality Criteria for Modal Logics

The discussion focuses on basic modal logic $\mathbf{K}$ and its extensions through common frame properties. All the concepts are easily generalisable to multi-modal logic $\mathbf{K}_{(m)}$ and its extensions. Due to space restriction, we do not formally define the syntax and semantics of modal logics. But for sake of clarity and to help the discussion in this short abstract, we recall the definition of a model for basic modal logic. A model $\mathfrak{M} = (W, R, V)$ is a triple where W is a non-empty set of worlds, R is the accessibility relations over W, and V is the labelling function that to each propositional symbol assigns a set of worlds. In

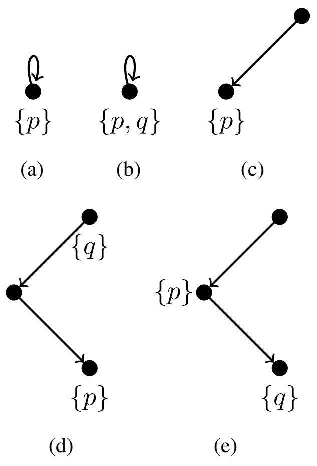


Figure 1: Possible models of $\Diamond p \lor (q \land \Diamond\Diamond p) \lor (\Diamond p \land \Box\Diamond q)$

the rest of the abstract, every time we refer to $\mathfrak{M}$ and $\mathfrak{M}'$ we mean two different models for the same modal formula such that $\mathfrak{M} = (W, R, V)$ and $\mathfrak{M}' = (W', R', V')$.

The first minimality criterion mentioned in the introduction is domain minimality. For modal logics the domain is represented by the set of worlds. This directly brings us to the following simple definition of domain minimality for modal logics. Given a model $\mathfrak{M}$, $\mathfrak{M}$ is a domain minimal model iff there is no model $\mathfrak{M}'$ such that $|W'| < |W|$, where $|W|$ represents the cardinality of W. Adopting this minimality criterion, (a) and (b) are the only domain minimal models among the models in Figure 1.

An important advantage in using domain minimality is that modal logics having the finite model property cannot have any infinite minimal model. On the other hand, creation of domain minimal model is achieved by trying to expand formulae in the scope of diamond operators in already existent worlds. Such expansion may be inefficient in practical implementations and may lead to minimal models with a questionable semantic meaning. For instance, what would be the meaning of the two domain minimal models in the figure if the resulting reflexive relation was something like has_child of $is\_married\_to$? In such cases another minimality criterion would be preferable.

In [8] we present a tableau calculus for the generation of minimal modal Herbrand models for the multi-modal logic $\mathbf{K}_{(m)}$ and its extensions with reflexivity and symmetry. A modal Herbrand model is a model obtained by expanding a diamond formula creating always a new successor. A possible definition of minimal Herbrand model is as follows. Given a model $\mathfrak{M}$, $\mathfrak{M}$ is a minimal modal Herbrand model iff for any other model $\mathfrak{M}'$, if $R' \subseteq R$ and $V'(p) \subseteq V(p)$ for all propositional symbol p, then $\mathfrak{M} = \mathfrak{M}'$. A requisite for comparing two modal Herbrand models is that the two sets of worlds are the same or that is possible to map one to the other, and vice versa. In [8] we obtain this by assigning unary functional symbols to diamond formulae. In Figure 1, (c), (d) and (e) are the three minimal modal Herbrand models of the formula under consideration.

Modal Herbrand models, as Herbrand models for first-order logic, are of interest for automated deduction systems. In fact, many semantic tableau methods for non-classical logics generate modal Herbrand models. For this reason, minimal modal Herbrand models may be seen as a natural minimality criterion for automated reasoning. Furthermore, the deterministic expansion of diamond formulae in exactly one successor solves the problem affecting domain minimality. The disadvantage of generating minimal modal Herbrand models is that the obtained models may be infinite. This implies that to ensure termination of a procedure generating such minimal models, the use of a blocking technique or particular closure tests are required. Blocking may result in models that are not completely Herbrand, because it can be thought as the application of a different diamond expansion when the model is infinite. A closure test to ensure termination may affect the completeness of the calculus.

As for domain minimality, it is possible to give a simple definition of the minimisation of a specific set of predicates as follows. Given a model $\mathfrak{M}$ and a set S of propositional symbols, $\mathfrak{M}$ is a minimal model iff for any other model $\mathfrak{M}'$ and all $p \in S$, if $V'(p) \subseteq V(p)$, then $V(p) = V'(p)$. Also in this case, the domains of the two models must be comparable as in the case of minimal modal Herbrand models. Assuming that $S = \{q\}$, then (a) and (c) are the two minimal model with respect to S among those in Figure 1.

Even though this kind of minimisation can be thought as a specific case of circumscription, it is worth to noting that circumscription does not require model generation but for minimal entailment reasoning. Circumscription for description logics has recently received some attention from both the complexity point of view ([1]) and the decision procedure point of view ([4]), and techniques used in circumscription may be a starting point for the generation of minimal model for specific sets of predicates.

The last criterion we take in consideration is minimality with respect to bisimulation. This is the only criterion of this abstract that to our knowledge does not correspond to an existing minimality criterion for classical logics. Given two models $\mathfrak{M}$ and $\mathfrak{M}'$, if $\mathfrak{M}$ is bisimilar to a submodel of $\mathfrak{M}'$, then $\mathfrak{M}$ is minimal with respect to $\mathfrak{M}'$. Using this criterion, all models in Figure 1 are minimal except the model (d). In fact, the model (d) is a supermodel of (c). It can be argued that also other models in the figure should not be considered minimal, such as (b) and (e). Their minimality is due to characteristics of bisimulation: two bisimilar worlds must agree on all the propositional symbols. This is why the model (b) is not considered a supermodel of (a), the zag condition of bisimulation requires a backward relation to preserve the accessibility relation of $\mathfrak{M}'$. This is why the model (e) is not considered a supermodel of (c). It is important to point out that bisimulation is the only presented criterion that is able to compare two models by changing the graph structures of the models under consideration (obviously the changes are such that the resulting model is equivalent to the originals).

3 Conclusion

We discussed several minimality criteria for modal logics. The presented criteria are not the only possible minimisations for modal logics. In fact, it is not difficult to think of a composition of them, or even completely new criteria. We believe that it is not possible to rank these minimality criteria and select "the best", such selection depends on what kind of constraints must be imposed to the models and what are the desired characteristics. This must, however, not stop the interest in creating formal procedures for the generation of minimal models for non-classical logics, as they can be useful in several areas of Computer Science including model checking and non-monotonic reasoning.

References

[1] P. A. Bonatti, C. Lutz, and F. Wolter. The complexity of circumscription in description logic. *J. Artif. Int. Res.*, 35:717–773, 2009.

[2] F. Bry and A. Yahya. Positive unit hyperresolution tableaux and their application to minimal model generation. *J. Automated Reasoning*, 25(1):35–82, 2000.

[3] L. Georgieva, U. Hustadt, and R. A. Schmidt. Computational space efficiency and minimal model generation for guarded formulae. In *Proc. LPAR'01*, volume 2250 of *LNAI*. Springer, 2001.

[4] S. Grimm and P. Hitzler. A preferential tableaux calculus for circumscriptive $\mathcal{ALCO}$. In *Proc. RR'09*, volume 5837 of *LNCS*, pages 40–54. Springer, 2009.

[5] S. Lorenz. A tableaux prover for domain minimization. *J. Automated Reasoning*, 13(3):375–390, 1994.

[6] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artif. Int.*, 13:27–39, 1980.

[7] I. Niemelä. A tableau calculus for minimal model reasoning. In *Proc. TABLEAUX'96*, volume 1071 of *LNCS*, pages 278–294. Springer, 1996.

[8] F. Papacchini and R. A. Schmidt. A tableau calculus for minimal modal model generation. *ENTCS*, 278(3):159–172, 2011.

Quantifier rules in reductive proof using nominal semantics

Murdoch J. Gabbay (reporting on work with Claus-Peter Wirth)

http://www.gabbay.org.uk

Abstract: Reductive proof-search tries to reduce a goal to tautologies. In the presence of quantifiers this becomes a complex design problem by which proof-theory shades into 'proof-engineering'. There is no single right answer here, but there are a lot of practical problems with strong roots in theory. In this work we consider a nominal semantics for this design space. The reduction in complexity is striking, and we get an elementary account of reductive proof-search with quantifiers.

1 The problem

It is not enough to study proof in principle; we may also want to prove things in practice. This means creating notions of derivation that are succinct and amenable to automation. We concentrate on *reductive*, *analytic*, or *backward* proof-search, reducing goals to assumptions. Examples include sequent, tableau, matrix, and indexed-formula-tree systems.

The quantifier $\forall$ is what interests us. It generates a pair of rules, called *left-intro* and *right-intro*, or $(\delta\forall)$ and $(\gamma\forall)$ rules respectively. Intuitively:

1. left-intro/$(\gamma\forall)$ means "$\forall x.\phi$ implies $[r/x]\phi$ for any r", where here $[r/x]$ is the usual capture-avoiding substitution of r for x (call r a **witness**).
2. right-intro/$(\delta\forall)$ means "$[r/x]\phi$ for some *sufficiently generic* r implies $\forall x.\phi$".

$(\gamma\forall)$ has obvious potential for branching and we delay the choice of witness as long as possible by introducing variables called *existential variables*, *meta-variables*, or (in tableaux) *free variables*. These are variables whose instantiation transforms the proof as a whole. We use variables X from nominal terms to do this (no surprise to the expert in nominal techniques; X was a unification unknown in [20]).

Concerning $(\delta\forall)$: what should 'sufficiently generic' mean? The standard rule takes a fresh entity variously called a *fresh constant*, a *fresh variable*, an *eigenvariable*, or *parameter*. Here it is in sequent style:

$$\frac{\Gamma \vdash \psi, \Delta \quad (x \text{ fresh for } \Gamma, \Delta)}{\Gamma \vdash \forall x.\psi, \Delta}\,(\forall\mathbf{R})/(\delta^{-}\forall)$$

This rule is inefficient because x is *unnecessarily generic*; choosing x 'completely fresh' does not record—and cannot take advantage of—information about what variables existed when x was created.

An industry exists devising rules to prove $\forall a.\phi$ more efficiently, and it is worthwhile to list some of it: Fitting's original free-variable δ-rule [7, Section 7.4]; then its 'liberalised' version δ^{+} (introduced in [16], and treated in [8, Section 7.4]); $\delta^{+^{+}}$ [2]; δ^{*} [1];[1] $\delta^{*^{*}}$ [3];[2] and δ^{ε} [15, Section 4.1].

[1]This had error corrected in [4, Subsection 5.3].

[2]This also had errors, also corrected in [4, Subsection 5.4].

So in the quest for efficiency, inference systems have developed interesting kinds of names and binding, and there is a direct connection between recognising how names in derivations interact (and when they must be generated, and when they may be thrown away) and devising efficient quantifier rules.[3] This kind of thing is hard to get right, errors have been made, and aside from work by Wirth reported in [21], no semantics has been available to aid understanding.

This abstract reports on cutting-edge research in the application of two nominal tools to reductive proof-search: *permissive-nominal terms* from [6] and *nominal sets* semantics from [14] (surveys in [9, 11]).

Technical details are elided. In this abstract we give a flavour of how this rather substantial body of mathematics hangs together. If pressed to describe this work in a sentence, it is this: we have an elementary explanation of the variables and meta-variables typically found in proof-search, as nominal atoms and unknowns, *and* of the proof-search rules listed above; and if you can get past the unfamiliar nominal-ness of the semantics, the technical difficulty threshold is quite low.

2 Sketch of the syntax

Fix disjoint countably infinite sets of **atoms** a, b, c and **variables/unknowns** X, Y, Z.

Atoms a are variables in goals and resemble the *parameters* or *eigenvariables* found in the literature. This is the entity introduced by the $(\forall\mathbf{R})$ rules of sequent systems. Variables X are proof-search variables; these display complex 'nominal' behaviour but have the effect of Skolem terms, without extending the signature or introducing functions. X corresponds to the *dummy* variable of [19] and [18] and the *free variable* of [8], the *meta-variable* of planning and constraint solving, and the *free γ-variable* of [21].

Assume constants $\bot$, $\top$, $\neg$, $\wedge$, and $\forall$ and a simple type system which we elide.

Then **terms** are just $r ::= a \mid X \mid \mathsf{f} \mid r'r \mid [a]r$.[4]

[3]Speedups can be significant. The (δ^{+}) of [16] allows exponential speedup relative to (δ^{-}) [8], and $(\delta^{+^{+}})$ [2] allows further exponential speedup relative to (δ^{+}) [1, Section 3].

[4]A white lie: X is moderated as $\pi{\cdot}X$. See [20, 6].

This looks familiar (variables; meta-variables; constants; application; abstraction) but substitution for X is capturing and the semantics of $[a]r$ is nominal atoms-abstraction instead of functional abstraction. So, the underlying semantics is different, non-functional, and 'first-order'; for details see [13] which applies this to Henkin-style semantics for higher-order logic.

Atoms are not constant symbols; models are subject to a permutation symmetry group of atoms, so atoms are special symmetric elements, translated specially in the denotation. Constant symbols are interpreted arbitrarily; no special properties are assumed. So $[a]r$ makes sense because the interpretation of a is specific such that atoms-abstraction has meaning; $[\mathsf{f}]r$ makes no model-theoretic sense because there is virtually no restriction on how f is interpreted.

3 Sketch of the proof-rules

Here are the two crucial rules, in sequent style:

$$\frac{\mathcal{C}, X{\uparrow}[a]\neg\phi; \mathcal{H} \vdash \phi[a{\mapsto}X]}{\mathcal{C}; \mathcal{H} \vdash \forall a.\phi}\ (\delta^{\mathsf{x}}\forall\mathbf{R}) \qquad \frac{\mathcal{C}; \phi[a{\mapsto}r] \vdash \psi}{\mathcal{C}; \mathcal{H}, \forall a.\phi \vdash \psi}\ (\delta^{\mathsf{x}}\forall\mathbf{L})$$

$\mathcal{H}$ is a set of predicates. $\mathcal{C}$ is a **maximisation** condition; a set of syntax of the form $X{\uparrow}[a]\phi$. This can be read as an instruction to 'maximise' the truth-value of $\phi[a{\mapsto}X]$. Intuitively, if the value of X makes $\neg\phi[a{\mapsto}X]$ true, then $\forall a.\phi$ must be true. If this reminds the reader of expressing $\forall a.\phi$ as $\phi[a{\mapsto}\epsilon a.\neg\phi]$ [5, page 15] then that is no accident—but here there is no choice made, only a maximisation *condition*.

The nominal semantics makes itself particularly useful because ϕ is a possibly open predicate—it may have free atoms. Nominal semantics allow us to map this open predicate to an open element of a nominal algebra of possibly open truth-values.

The underlying message is that nominal semantics here replace Skolemisation in both syntax and semantics. The price we pay is a notion of 'truth-values algebra with atoms', but this seems not only worthwhile but is in itself interesting.

4 Sketch of the semantics

We just give the flavour of how it works; the particularly dedicated reader can find full details of similar technology applied in abstract algebra in [10]. We assume a Boolean algebra, but elements are nominal and contain free atoms. These atoms can be substituted for; *this substitution is not syntactic*, but an abstract nominal algebraic axiomatisation of substitution following [12]. Next is quantification, which is just an operation on algebra elements related to the quantification operation of *cylindric algebra* [17] but in a nominal context. Atoms are interpreted as themselves and variables X are interpreted with valuations.

Once all this is in place, proving compositionality of the semantics *and* the proof-rules is very easy, because the semantics includes abstract nominal structures which mirror what is done in the syntax. Every 'normal' model (with Skolemisation and choice) can be converted to a nominal model, so this simplicity does not come by absurd restriction of models.

5 Summary

What we have discussed can be accomplished with choice and Skolemisation. But these are powerful, and using such tools has a price; we must manipulate a system with more structure than necessary, and must use many emulations.

Nominal techniques give the benefits of Skolemisation without introducing functions or higher types. Maximisation conditions give the benefit of Hilbert's choice without making choices. The nominal semantics is not hard—we can even import it off-the-shelf from [13, 10]—and allows us to talk about open elements easily and directly, and it all fits together nicely.

References

[1] M. Baaz and C. G. Fermüller. Non-elementary speedups between different versions of tableaux. In *Proceedings of the 4th International Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX'95)*, pages 217–230. Springer, 1995.

[2] B. Beckert, R. Hähnle, and P. H. Schmitt. The even more liberalized delta-rule in free variable semantic tableaux. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Proceedings of the third Kurt Gödel Colloquium (KGC'93)*, volume 713 of *LNCS*, pages 108–119. Springer, 1993.

[3] D. Cantone and M. Asmundo. A further and effective liberalization of the δ-rule in free variable semantic tableaux. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, pages 408–414. Springer, 2000.

[4] D. Cantone and M. Nicolosi-Asmundo. A sound framework for delta-rule variants in free variable semantic tableaux. *Journal of Automated Reasoning*, 38:31–56, 2007.

[5] P. B. David Hilbert. *Grundlagen der Mathematik (volume II)*. Number 50 in Grundlehren der mathematischen Wissenschaften. Springer, second edition, 1970.

[6] G. Dowek, M. J. Gabbay, and D. P. Mulligan. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[7] M. Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 1 edition, 1990.

[8] M. Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 2 edition, 1996.

[9] M. J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[10] M. J. Gabbay. Stone duality for First-Order Logic: a nominal approach. In *Howard Barringer Festschrift*. December 2011.

[11] M. J. Gabbay. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*, volume 17. Kluwer, 2012.

[12] M. J. Gabbay and A. Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.

[13] M. J. Gabbay and D. Mulligan. Nominal Henkin Semantics: simply-typed lambda-calculus models in nominal sets. In *Proceedings of the 6th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2011)*, volume 71 of *EPTCS*, pages 58–75, September 2011.

[14] M. J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, July 2001.

[15] M. Giese and W. Ahrendt. Hilbert's ϵ-terms in automated theorem proving. In N. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes in Computer Science*, pages 662–662. Springer, 1999.

[16] R. Hähnle and P. H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–222, 1994.

[17] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras*. North Holland, 1971 and 1985. Parts I and II.

[18] S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 364–371. Springer, 1963.

[19] D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 159–199. Springer, 1983.

[20] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, September 2004.

[21] C.-P. Wirth. Descente infinie + Deduction. *Logic Journal of the IGPL*, 12(1):1–96, 2004.

Model Checking by Abstraction for Proving Liveness Properties of Hybrid Dynamical Systems

Rebekah Carter   Eva M. Navarro-López

School of Computer Science
The University of Manchester
{carterr,eva.navarro}@cs.man.ac.uk

Abstract: In this extended abstract, we consider a method for model checking certain properties of continuous dynamical systems, by adapting an abstraction method previously proposed so that it will work with the timed automata model checker UPPAAL. We also describe our ongoing work to extend this method to verify liveness (reachability) properties of hybrid dynamical systems.

1 Introduction

Hybrid systems are very complex, characterised by interactions between continuous and discrete parts. For instance, a controlled robot can be modelled as a hybrid system: it has continuous dynamics governing *how* it moves in the real-world, and discrete behaviour of the controller making decisions on *what* it sets out to do.

In the academic community, a lot of effort has gone into trying to verify such hybrid systems, mostly through using model checking methods. One class of such methods involves abstraction of the hybrid system to a discrete system (usually a finite-state automaton) in order to be able to use model checkers on these finite automata to prove properties. The properties that can be proved are mostly safety properties of the form $\Box\phi$, saying that we always satisfy some 'safe' predicate ϕ, and in fact these finite automaton abstractions are unable to prove liveness-type properties, even the most simple reachability-type $\Diamond\psi$.

In order to prove these reachability-type properties, we need to keep some information about the times at which events occur in a discrete system, and so we turn to using timed automata (TAs) for our abstraction. In Section 2 we consider a method by Maler and Batt [2] which abstracts a continuous system to a TA, making use of multiple clocks. We discuss interfacing this algorithm with the most commonly used TA verification program, UPPAAL [1], in Sect. 3. We describe our current and future work in Sect. 4.

2 The Algorithm of Maler and Batt

In [2], Maler and Batt define an algorithm to approximate continuous systems by TAs. The approximation method they use is intuitive, based on minimum and maximum velocities of a system defining bounds on the time taken to cross a certain distance in the system. This creates an over-approximation of the system, in the sense that every trajectory in the system is matched by one in the abstraction, but additional trajectories may be allowed in the abstraction.

The basic idea is to split the state space of a continuous system into hyper-cubes of unit length, and to define clocks to keep track of the time crossings that are made in each

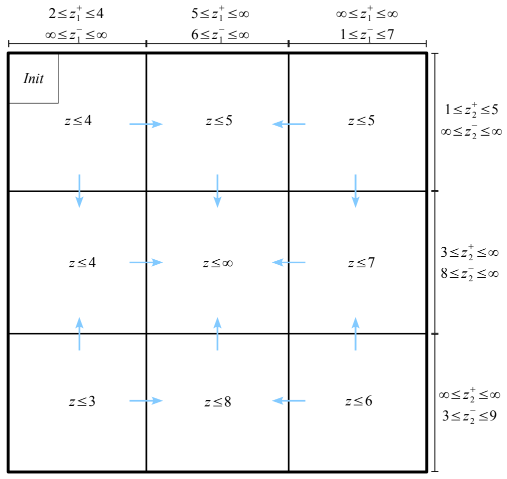


Figure 1: Split the state space into hyper-cubes (rectangles in this case, as system in $\mathbb{R}^2$), calculate which edges exist (denoted by arrows) and calculate the minimum/maximum times on the clocks z, z_1^+, ...

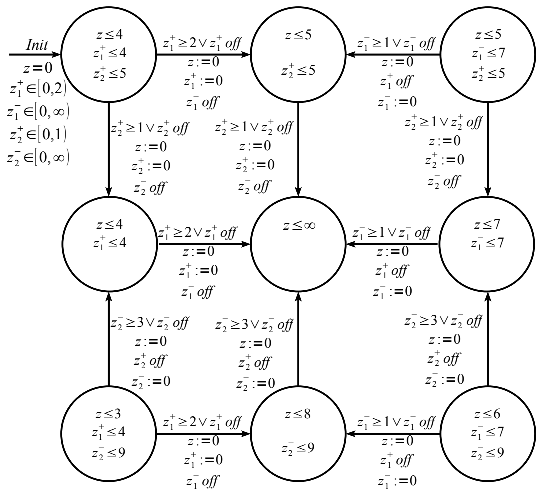

Init
$z=0$
$z_1^+\in[0,2)$
$z_1^-\in[0,\infty)$
$z_2^+\in[0,1)$
$z_2^-\in[0,\infty)$
$z\le 4$ $z_1^+\le 4$ $z_2^+\le 5$
$z_1^+\ge 2\vee z_1^+$ *off* $z:=0$ $z_1^+:=0$ z_1^- *off*
$z\le 5$ $z_2^+\le 5$
$z_1^-\ge 1\vee z_1^-$ *off* $z:=0$ z_1^+ *off* $z_1^-:=0$
$z\le 5$ $z_1^-\le 7$ $z_2^+\le 5$
$z_2^+\ge 1\vee z_2^+$ *off* $z:=0$ $z_2^+:=0$ z_2^- *off*
$z_2^+\ge 1\vee z_2^+$ *off* $z:=0$ $z_2^+:=0$ z_2^- *off*
$z_2^+\ge 1\vee z_2^+$ *off* $z:=0$ $z_2^+:=0$ z_2^- *off*
$z\le 4$ $z_1^+\le 4$
$z_1^+\ge 2\vee z_1^+$ *off* $z:=0$ $z_1^+:=0$ z_1^- *off*
$z\le\infty$
$z_1^-\ge 1\vee z_1^-$ *off* $z:=0$ z_1^+ *off* $z_1^-:=0$
$z\le 7$ $z_1^-\le 7$
$z_2^-\ge 3\vee z_2^-$ *off* $z:=0$ z_2^+ *off* $z_2^-:=0$
$z_2^-\ge 3\vee z_2^-$ *off* $z:=0$ z_2^+ *off* $z_2^-:=0$
$z_2^-\ge 3\vee z_2^-$ *off* $z:=0$ z_2^+ *off* $z_2^-:=0$
$z\le 3$ $z_1^+\le 4$ $z_2^-\le 9$
$z_1^+\ge 2\vee z_1^+$ *off* $z:=0$ $z_1^+:=0$ z_1^- *off*
$z\le 8$ $z_2^-\le 9$
$z_1^-\ge 1\vee z_1^-$ *off* $z:=0$ z_1^+ *off* $z_1^-:=0$
$z\le 6$ $z_1^-\le 7$ $z_2^-\le 9$

Figure 2: The timed automaton resulting from the method of [2] applied to the dynamical system. If a clock is not given an explicit bound in a location, it is *clock* $\le\infty$.

dimension. We then put bounds on these clocks within each cube which limit the amount of time we can spend in a cube or a slice of the space (made up of cubes). Other clock bounds also limit the times when a transition can be taken, capturing extra information about the system.

Figures 1–2 show the process of making an abstraction of the system by this method.

3 Changes to Use UPPAAL for Verification

UPPAAL [1] is the most widely used tool for modelling and verification of TAs, so interfacing it with the method of [2] will increase the number of people who can make use of it. UPPAAL has a few restrictions for this method:

1. UPPAAL does not accept disjunctions of conditions in the guards of edges of the TA.
2. UPPAAL defines all clocks to start at value zero.
3. UPPAAL must have integer valued comparison values for clocks.

3.1 Removing Disjunctions

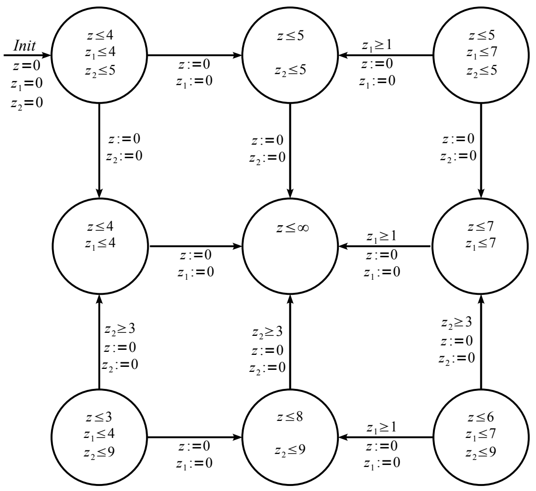
For the method of Maler and Batt, it is necessary to have disjunctions in the invariants of the locations and in the guard conditions of the edges in the TA. As UPPAAL will not allow this we must remove the necessity of checking whether clocks are active, and we achieve this by only having one clock in each dimension, plus the one which deals with time spent in cubes. To preserve the approximation, when exiting a slice of the space in the positive (negative) direction, we only allow minimum time guards to be present if the flow in this slice was always positive (negative).

3.2 Removing Set-Valued Initial Conditions

The method of [2] requires that the initial condition on the clocks be set-valued, in order to allow the abstracted timed automaton to start anywhere within an initial cube. UPPAAL only allows clocks to start at zero, so to preserve over-approximation with respect to trajectories, one option is to not allow minimum times on clocks to be specified anywhere in the slice in which an initial cube appears. Another option we are considering is to simulate the set-valued initial condition in UPPAAL, using non-deterministic transitions to allow various initial clock values.

3.3 Scaling Time to the Order of the Integers

As UPPAAL only allows clocks to be compared to integers, it is necessary (once the TA is calculated) to scale the minimum/maximum time bounds to roughly the order of the integers. This is seen particularly clearly in electronic circuit examples, where the scale of time is typically 10^{-4}. The main step in this rescaling is to divide all time bounds by the minimum value, so that the minimum becomes 1 and other bounds are greater than this. We must also take the ceiling (resp. floor) of the maximum (resp. minimum) times to give integer comparison values.



Init
$z=0$
$z_1=0$
$z_2=0$
$z\leq 4$ $z_1\leq 4$ $z_2\leq 5$
$z\leq 5$ $z_2\leq 5$
$z\leq 5$ $z_1\leq 7$ $z_2\leq 5$
$z:=0$ $z_1:=0$
$z_1\geq 1$ $z:=0$ $z_1:=0$
$z:=0$ $z_2:=0$
$z:=0$ $z_2:=0$
$z:=0$ $z_2:=0$
$z\leq 4$ $z_1\leq 4$
$z\leq\infty$
$z\leq 7$ $z_1\leq 7$
$z:=0$ $z_1:=0$
$z_1\geq 1$ $z:=0$ $z_1:=0$
$z_2\geq 3$ $z:=0$ $z_2:=0$
$z_2\geq 3$ $z:=0$ $z_2:=0$
$z_2\geq 3$ $z:=0$ $z_2:=0$
$z\leq 3$ $z_1\leq 4$ $z_2\leq 9$
$z\leq 8$ $z_2\leq 9$
$z\leq 6$ $z_1\leq 7$ $z_2\leq 9$
$z:=0$ $z_1:=0$
$z_1\geq 1$ $z:=0$ $z_1:=0$

Figure 3: The TA created by the revised method.

3.4 The Revised Method

Figure 3 shows a revised TA for the same system as Figs. 1–2, using the new aspects of the method discussed here. The important result of [2], which is preserved here, is that the abstraction is *neo conservative*. This means that every trajectory of the actual continuous system can be matched in time and location by a trajectory of the TA.

4 Ongoing Work

This method is able to deal with any continuous system as an abstraction technique, and it can be used in an abstraction refinement loop to keep improving the size of the abstraction until a desired property is proved. We have currently implemented this method for linear continuous systems with liveness (reachability) properties.

We are extending this method to deal with piecewise continuous systems, and also hybrid systems (modelled as hybrid automata). The main problem we forsee is when dealing with resets (the extra part in hybrid systems), and it may be necessary to restrict the type of resets we allow.

Acknowledgements

Both authors acknowledge the support of the EPSRC, under the framework of the project *DYVERSE: A New Kind of Control for Hybrid Systems* (EP/I001689/1).

References

[1] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In *SFM-RT'04*, volume 3185 of *LNCS*, pages 200–236, 2004.

[2] Oded Maler and Gregory Batt. Approximating Continuous Systems by Timed Automata. In *Formal Methods in Systems Biology*, volume 5054 of *LNCS*, pages 77–89, 2008.

Verification of nonpolynomial systems using MetiTarski

William Denman

Computer Laboratory, University of Cambridge, England
william.denman@cl.cam.ac.uk

Abstract: Complex systems generally do not admit closed form solutions, requiring the use of potentially inaccurate numerical methods for their analysis. On the other hand, qualitative methods such as Lyapunov functions, do not require explicit solutions but can be extremely difficult to use in practice. Preliminary results show that the automated theorem prover MetiTarski is useful in the analysis of hard verification problems such as the reachability, stability and abstraction of continuous and hybrid nonpolynomial systems.

1 Introduction

Formal verification is a broad field that encompasses a number of mathematical and logical techniques for proving the correctness of models. Until recently, automated formal verification research has been focused on discrete-state systems. There have been several breakthroughs and now formal methods are beginning to find a place in industrial work-flows. Examples include the verification of software and digital circuits. On the other hand, there are still not many options for the automated formal verification of continuous systems. Techniques that do exist are either constrained to linear systems or can only be applied to polynomial nonlinear systems.

MetiTarski [2] is an automated theorem prover for arithmetical conjectures involving transcendental functions (sin, cos, exp etc.). It has been successful in proving arithmetical theorems that are used to verify the behaviour of certain analogue circuits [3] and linear hybrid systems [1]. The next step is to leverage the power of MetiTarski for the analysis and verification of continuous and hybrid systems described by transcendental and special functions.

The input to MetiTarski is a first-order formula that may involve inequalities that contain transcendental functions. The question is then : How can a complex engineering verification problem be translated into a sequence of MetiTarski problems? If the behaviour of the system can be described using linear differential equations, then an analytic solution (trajectory) can be computed and will contain transcendental and special functions. It will then be possible to reason directly about whether the system will reach an unwanted state using MetiTarski.

Most engineering systems can only be specified using nonlinear differential equations and closed form solutions are generally not computable. This is because, not surprisingly, nonlinear systems present a richer dynamics than purely linear systems. It is for these reasons that the analysis of nonlinear dynamics uses a combination of both qualitative analysis and repeated numerical simulation [6]. However, any numerical method can potentially be incorrect due to the finite precision used to calculate the result.

2 Abstraction of Continuous and Hybrid Systems

Safety, the fact that some bad behaviour will never happen, is perhaps the most important property that should be verified for a system. The reachability computation remains the most common way to check safety of a system. However, it cannot be stressed enough that the exact reachability of continuous and hybrid systems cannot be computed. Therefore most verification methods compute an over-approximation of the reachable states to enable a decidability result [4]. Another equally important method is abstraction, which reduces the complexity of the system but at the same time must preserve the properties under verification.

By abstracting properly and preserving the relevant underlying behaviour of the system, tools that are already developed can be used. Sloth and Wisniewski [7] developed a method for creating a sound and complete abstraction of continuous systems using Lyapunov functions. By using the Lyapunov function as a predicate for partitioning, they were able to convert the infinite state space of a continuous system into timed automata.

There are several restricted classes of hybrid systems and continuous systems that have been shown to have decidable reachability properties but most are too weak for practical applications. Another method of abstraction borrows from the domain of qualitative reasoning. Qualitative reasoning is motivated by the idea that numerical simulation is limited when not all the parameters of the system are known. Instead of trying to compute a solution, it is sufficient to look at how the vector field itself changes over time. Tiwari [9] uses predicates that evaluate over the three symbols $\{+, -, 0\}$ to split up the infinite state space. This construction of the abstraction uses the decidability of the first order theory of real closed fields [8] to compute the transitions between abstract states. Once the abstraction is created then a model checker is used to evaluate CTL properties on the abstract system. For linear systems it is easy to choose the predicates, but nonlinear systems still require a search that relies mostly on heuristics.

3 Reachability : Aircraft Collision Avoidance

In air traffic control, collision avoidance protocols are used to direct planes in case of a catastrophic error. This could be caused by the pilots themselves, flight directors on the ground, faults within the air control software or physical system faults. The avoidance protocols must be able to make the correct decision quickly because at the point that a possible collision is detected, there might be at most one minute of time to react [5].

If we consider a single aircraft flying in the xy plane the following system of differential equations describes its behaviour

$$\begin{aligned} x_1'(t) &= d_1(t) & x_2'(t) &= d_2(t) \\ d_1'(t) &= -wd_2(t) & d_2'(t) &= wd_1(t) \\ x_1(0) &= x_{1,0} & x_2(0) &= x_{2,0} \\ d_1(0) &= d_{1,0} & d_2(0) &= d_{2,0} \end{aligned}$$

with $x_1'(t)$ and $x_2'(t)$ the speed in the x and y direction. $d_1'(t)$ and $d_2'(t)$ the acceleration in the x and y directions. $d = (d_1, d_2) = (v\cos(\theta), v\sin(\theta))$ a linear speed vector that defines both speed and orientation together and w the angular velocity.

Solving for two aircraft travelling in a plane $((x_1, x_2), (y_1, y_2))$ gives us the following set of positional equations.

$$x_1(t) = x_{1,0} + \frac{d_{2,0}\cos(wt) + d_{1,0}\sin(wt) - d_{2,0}}{w}$$

$$x_2(t) = x_{2,0} + \frac{d_{1,0}\cos(wt) - d_{2,0}\sin(wt) - d_{1,0}}{w}$$

$$y_1(t) = y_{1,0} + \frac{e_{2,0}\cos(wb\,t) + e_{1,0}\sin(wb\,t) - e_{2,0}}{wb}$$

$$y_2(t) = y_{2,0} + \frac{e_{1,0}\cos(wb\,t) - e_{2,0}\sin(wb\,t) - e_{1,0}}{wb}$$

To ensure that the planes do not hit each other, it is required that they keep a minimum separation distance p. This can be described mathematically as

$$(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$$

We have been able to prove this separation using MetiTarski for two aircraft in a plane flying at different but constant angular velocities. This simplification reduces the problem to one variable. A collision avoidance protocol would have varying velocities and therefore would add more variables to the resulting problem. Recent improvements to MetiTarski have made these multi-variable problems more tractable.

4 Conclusion

The verification problem of nonpolynomial systems remains wide open. We are currently working on several methods that use MetiTarski to construct valid abstractions of nonpolynomial vector fields. Abstraction based methods are essential because analytical solutions to nonlinear systems normally do not exist and in some cases numerical methods diverge quickly. There are verification techniques for abstracted nonlinear polynomial hybrid systems but none for systems described using transcendental and other special functions.

Beyond the idea of merely creating a method for verifying hybrid systems, we want to be able to verify real world engineering problems. This will require that any techniques we develop are able to scale properly to larger systems. Abstraction techniques can be scaled as long as it is possible to calculate correct transitions in the abstract system. Formal tools for the analysis of discrete state systems would then be directly applicable to the abstract models.

References

[1] B. Akbarpour and L. C. Paulson. Applications of MetiTarski in the verification of control and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 1–15, 2009.

[2] B. Akbarpour and L. C. Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44:175–205, 2010.

[3] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson. Formal verification of analog designs using MetiTarski. In *Formal Methods in Computer-Aided Design. FMCAD 2009*, pages 93 –100, November 2009.

[4] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[5] A. Platzer. *Logical Analysis of Hybrid Systems*. Springer, 2010.

[6] S. Sastry. *Nonlinear Systems*. Springer, 1999.

[7] C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing lyapunov functions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3760–3765, December 2010.

[8] A. Tarski. A decision method for elementary algebra and geometry. Technical report, RAND Corp., 1948.

[9] A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *Hybrid Systems: Computation and Control HSCC*, volume 2289 of *LNCS*, pages 465–478. Springer, Mar. 2002.

A Formal Behaviour Representation for the Analysis of Distributed Simulations of Unmanned Aircraft Systems

Ariane Piel

ONERA–The French Aerospace Lab, 91123 Palaiseau, France, Ariane.Piel@onera.fr

Abstract: This thesis, devoted to activity recognition, sets itself within the framework of distributed simulation data analysis, relying on a particular temporal logic, the chronicle language. The aim is to model the notion of chronicle recognition and to prove the validity of this model in the perspective of developing a tool for analysis.

1 Introduction

This work tackles activity recognition in complex problems. The first applications explored in this field were related to the configuration of alarms in planes (defining which alarms should be transmitted to the pilot). Other applications concern notably informatics security.

In these situations, several agents interact through time, and it is difficult, not to say impossible, to proceed to a real experiment, especially for aerospace systems since they are very critical. It is therefore necessary to carry out distributed simulations. These produce a huge mass of data on which deduction must be assisted in order to detect hazardous behaviours.

The chronicle language has been introduced by C. Dousson in [4] and P. Carle in [2] to describe arrangements of events. In his Ph.D. [1], O. Bertrand formalises the notion of chronicle in order to model their recognition in an event flow using coloured Petri nets. This model was reworked in [3] by P. Carle, C. Choppy and R. Kervarc. In the continuation of this work, we propose to rigorously formalise all of the notions and tools used in order to then show the validity of the recognition system, and apply it to the surveillance and verification of Unmanned Aircraft Systems (UAS), whether it is to manage failures or to prevent collision risk as much as possible.

2 Chronicles

The chronicle language. The chronicle language has been developed in order to formally describe behaviours within a flow of events. It is inductively defined as follows:

$$C ::= A \mid C\,C \mid C\&C \mid C \mid\mid C \mid (C) - [C]$$

where the operators respectively correspond to a single event, a sequence, a conjunction, and an absence.

The stake is to recognise the instances of a chronicle in an event flow (*i.e.* a sequence of events).

To achieve this, the notion of chronicle recognition is formalised by inductively defining the set $R_C(\varphi)$ of the recognitions of chronicle C in flow φ.

Coloured Petri net model. In order to model chronicle recognition, a model using coloured Petri nets has been developed. For any chronicle C, an associated coloured Petri net $N(C)$ is built by induction. Irrespective of chronicle C, $N(C)$ has the same global structure presented in Figure 1: place Start(C) plays a role in the chronicle sequence, place Present(C) contains the value of the event meter, place Wini(C) is used for absences, and place Success(C) contains the list of recognitions of C. This common structure allows a linear and inductive construction which then makes it possible to formally reason on the nets and prove the validity of the model.

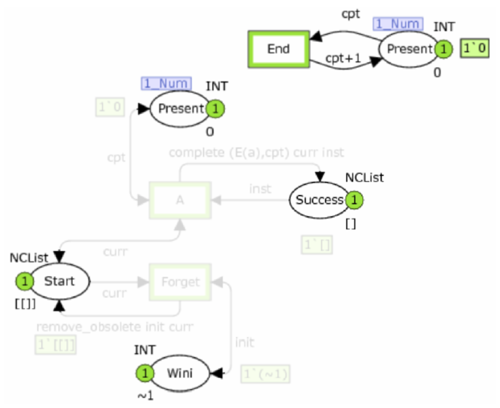


Figure 1: Structure of the nets associated to chronicles

Validity of the model. Hence, for each chronicle C, a function which associates to an event flow φ the set $R_C(\varphi)$ of the recognitions of C in this event flow has been defined (*recognition*). In order to model chronicle recognition, a Petri net $N(C)$ has been built associated to each chronicle C (*translation*). According to the event flow φ, the marking (M_i) of this net evolves (*marking*), and it is thus possible to read in it the recognitions of chronicle C in event flow φ (*interpretation*).

This system is represented by this diagram:

$$\begin{tikzcd}
C \arrow[r, "recognition"] \arrow[d, "translation"'] \arrow[dr, phantom, "\circlearrowleft"] & \varphi \mapsto R_C(\varphi) \\
N(C) \arrow[r, "marking"'] & \varphi \mapsto (M_i) \arrow[u, dotted, "interpretation"']
\end{tikzcd}$$

In the following theorem, the commutativity of the diagram, or in other words the fact that the recognitions read in the Petri nets correspond exactly to the theoretical definition of $R_C(\varphi)$, is established.

Theorem. *Let φ be an event flow. Let C be a chronicle and $N(C)$ the associated Petri net. After having fired the transitions associated to φ, the marking of place* `Success`(C) *corresponds exactly to $R_C(\varphi)$.*

Chronicle Recognition Library (CRL). In addition to the coloured Petri net model, a program has been developed in C++ in order to directly analyse online distributed simulation data. Its algorithms straightforwardly result from the theoretical set definitions of chronicle recognition.

3 Application to Unmanned Aircraft Systems (UAS)

The Unmanned Aircraft System. An unmanned aircraft system (UAS) is composed of three entities, the unmanned aircraft (UA), the remote pilot station (RPS), and the air traffic control (ATC). All three interact via several communication links as represented in Figure 2. This application deals with the problem of certifying UAS in order to insert them in controlled or uncontrolled airspace. The dynamic data flows between the agents of the system and between different systems if several UAS are considered are very complex. In addition, each agent deduces from its own observations the state of the other agents. Therefore, chronicles are well-suited to address the situation.

UAS operation safety analysis in case of fault. In [5], the faults that may arise within the system have been studied, and scenarios have been established to codify the behaviour to follow and thus ensure the security of the system and its environment even in critical situations. In this work, the synchronisation of the different elements of the system during breakdown situations has been examined, with, as a first step, telecommand failure. To this end, a state-transition UML diagram describing the procedure to follow has been developed. It has then been implemented in C++ with the help of the MSM (Meta State Machine) library of boost. Chronicles have then been determined to oversee the behaviours of the unmanned aircraft (UA), the remote pilot station (RPS), and the air traffic control (ATC).

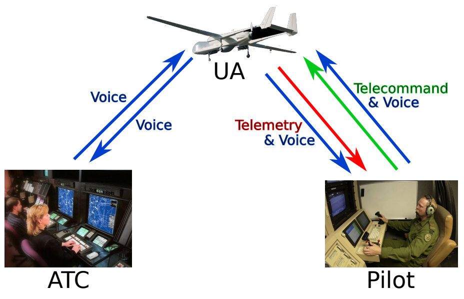


Figure 2: Structure and links within an UAS

4 Conclusion, Ongoing Work, and Perspectives

In conclusion, a well-established formal framework together with a Petri net model have been developed. The latter provides an acute understanding of the mechanisms of chronicle recognition, and the C++ library enables direct analysis of simulation data. The application to UAS tackles a problem of particularly great importance in the field of civil aviation and shows that chronicles are an interesting means to address this kind of issue.

The work presented here deals with theoretical foundations and practical applications, the two of which are to be pursued. Chronicles and both associated recognition models shall be extended to improve the expressivity of the framework. Delay-related constructs inspired from interval logics are currently considered for inclusion. The UAS application previously exposed tackles the crucial problem of the insertion of UAS in controlled or uncontrolled airspace and the associated certification and safety analysis processes. This first step deals with physical failures in one aircraft but there are other issues, as separation and collision avoidance which are the cornerstone of flight safety. A model for this is currently being developed in collaboration with UAS engineers. Chronicles, already applied to a wide variety of different fields, are shown to be an adequate generic means to represent knowledge in a multi-agent system, and, as such, benefit to a large spectrum of applications. Moreover, the insertion of UAS into general airspace raises concerns that cannot be solved without a formal representation which chronicles seem to fulfill fairly.

Acknowledgements

This work is done under the supervision of R. Kervarc[1], C. Choppy[2] and P. Carle[1]. The UAS application benefits from useful insights of T. Lang[1] and C. Le Tallec[1].

1: ONERA–DCPS 2: LIPN, U. Paris 13 & CNRS

References

[1] O. Bertrand. *Détection d'activités par un système de reconnaissance de chroniques et application au cas des simulations distribuées HLA* (PhD). U. Paris 13, 2009.

[2] P. Carle, P. Benhamou, F.-X. Dolbeau, and M. Ornato. La reconnaissance d'intentions comme dynamique des organisations. In *Proc. 6es JFIADSMA*, 1998.

[3] Patrice Carle, Christine Choppy, and Romain Kervarc. Behaviour recognition using chronicles. In *Proc. 5th IEEE International Symposium on Theoretical Aspects of Software Engineering*, p. 100–107, 2011.

[4] C. Dousson, P. Gaborit, and M. Ghallab. Situation Recognition: Representation and Algorithms. In *Proc. Int. Joint Conf. Artificial Intelligence*, 1993.

[5] Thibault Lang. IDEAS–T1.1: Architecture de système de drone et scénarios de missions. Technical report, ONERA–The French Aerospace Lab.

Analysing Data-Sensitive and Time-Sensitive Web Applications

Mohammed Alzahrani   Lilia Georgieva

Department of Computer Science,
Heriot-Watt University, Edinburgh, EH14 4AS, UK
{mya9,lilia}@hw.ac.uk

Abstract: Modelling and analysing *data-sensitive* and *time-sensitive* web applications allows designers to find security flaws, eliminate navigation errors, and simplify the applications' design. We study the use of the model checker SPIN for analysis of navigation and security properties. In this paper the web application is modelled as a two-timed finite state machine representing the web pages and the internal states. Our model is realistic; we augment it by introducing discrete time, web-session management, and authentication.

1 Introduction

Web applications have become common in today's economic and social life. Sectors like banking, shopping and governmental services are relying on the internet to promote and deliver their services [11, 8]. Such applications have a time-dependent behaviour and need to be secure due to, for example, their distribution and complexity aspects. The data that web applications handle is often sensitive (e.g. credit card numbers) to both the users and the service providers. Web applications' vulnerabilities, which may lead to the compromise of sensitive information are being reported continuously (See for example [9]).

As a result, new methods to support the design, implementation and verification of web applications need to be applied. A successful verification technique is model checking [2]. Model checking provides a fully automatic analysis and verification of the initial system design; by constructing a model of a system and its desirable behaviour a model checker can carry out a verification task. If an error is found, the tool shows under which circumstances the error can be generated. Traditionally model checking has been widely used for verification of hardware systems, more recently model checkers have been increasingly used for the verification of communication, security protocols and in software development [2, 7].

2 Modelling Web Applications

We build a formal model of a web application and investigate its possible behaviour under different scenarios. The desirable behaviour of the web application is written as a formal requirement specification. In order to address data and time sensitivity, we model both the *time* and the specific security protocol used (in one of our case studies SSL 3.0). By integrating discrete time in our model, we initially simulate a security protocol at the start of the session and users' authentication.

Security protocols (such as SSL 3.0) are widely used in online banking to ensure that the online data transmission is secure.

When modelling and verifying distributed applications capturing time-bound message exchange is important. For example, when a message between parties does not arrive within pre-fixed time, this will result in *timeout*, data retransmissions or other actions have to be considered at this stage [3].

We model web applications as concurrent systems where only one component executes an action at a time and the concurrent actions are arbitrarily ordered. We are interested in the verification of the application behaviour properties, rather than data transmitted properties. However, we model both web pages transactions and the business logic of a web applications, since the nature of web applications is dynamic and different input could lead different pages (i.e. wrong authentication credentials).

We model web applications models using a finite state automata. The use of automata models allow us to employ a clear formalism, which can be used in analysis and verification [7]. Our model uses two finite state automata and extends the work presented in [8]. The first automata specifies transitions between web pages, the second represents the internal state transitions (business logic) of the web applications.

Time in our model is discrete. Using discrete time adds realism to web applications' models. Modelling using discrete time allows us to both represent a web session management and to simulate timeout scenario in a web application. In model checking timed models, discrete time is preferred as it reduces the risk of *the state space explosion* [13].

3 SPIN and Promela

SPIN and its input language Promela developed by [7] for the verification and analysis of distributed software systems making them suitable for simulating and verifying web applications. The models written in Promela (a Process Meta Language), demonstrate the system components as processes that can communicate between each other via channels, earthier by buffered message exchange or rendezvous operations, and also through shared memory represented as global variables [1, 7]. As a model checker SPIN has been successfully applied in simulating and formal verifying security protocols and web applications. Since SPIN does not support the modelling of *time*, we extend our model with discrete time macros, similarly to [1].

4 Our model

We first use Promela and SPIN [7] to express the navigation properties of a standard stand-alone web application (e.g. online banking). We next verify the correctness of the web application by comparing compromised and secure models.

The following are examples of navigation properties: (i) The *h*ome page is reachable from all pages. A user can logoff at any stage of the transition. (ii) The *a*ccount page is reachable from all pages. (iii) A page reachable from the *h*ome page or *a*ccount page always has a next page in the transition.

Example of security properties are: (i) At a time t_1 only one user u_1 can be authenticated with the same password or user name. (ii) A user must interact with the web application within a pre-set time limit. (iii) The web application must receive the unaltered user's data as send by the user.

We model both secure and a compromised user interaction with a web application. To analyse to data exchange, we first develop a secure model, where the user of the web application authenticates herself correctly and responds within the required pre-set time limit.

Next, we introduce an attacker which interferes with the communication at various stages (week points) in the message exchange, for example by intercepting and altering the message exchange.

We use discrete time to analyse and compare the models and their simulation. In a compromised model, for example, if the attacker is active and attempts to impersonate the client, the sequence of actions modelled by the time stamp is different than the intended one. For example, during a man in the middle attack the compromised model is more complex than the secure one and the sequences of actions is different than the intended one. In a denial of service attack, for example, there may be a delay in the client accessing her bank account.

5 Related Work

A related approach for modelling web applications is developed in [6]. The approach uses communicating finite automata models which are based on the user-defined properties that are to be validated. Similarly to our approach, in [5, 12] web applications are modelled as pages, links and frames in one state transition.

An alternative framework for modelling of high-level specification of interactive, data-driven web applications is proposed in [4]. The work specifies data-driven web application and is not concerned with security or timeliness issues. Similarly, behaviour and structure of web applications are modelled in [10]. Test cases for the structure and behaviour of the applications are then derived automatically.

Purposefully developed tools, such as ReWeb have been used to analyse web applications [5]. In contrast, we do not rely on specially developed tools, but use a standard model checker.

References

[1] D. Bosnacki, D. Dams. Integrating real time into spin: A prototype implementation. In *FORTE XI / PSTV XVIII*, pages 423–438. Kluwer, B.V., 1998.

[2] Edmund M. Clarke. The birth of model checking. In *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 1–26, 2008.

[3] R. Corin, S. Etalle, P.H. Hartel, and A. Mader. Timed analysis of security protocols. *Journal of Computer Security*, 15(6):619–645, 2007.

[4] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web applications. *Journal of Computer and System Sciences*, 73(3):442–474, 2007.

[5] E. Di Sciascio, F.M. Donini, M. Mongiello, and G. Piscitelli. Web applications design and maintenance using symbolic model checking. In *7th European Conf on Software Maintenance and Reengineering. Proc.*, pages 63–72. IEEE, 2003.

[6] M. Haydar, A. Petrenko, and H. Sahraoui. Formal verification of web applications modeled by communicating automata. *Formal Techniques for Networked and Distributed Systems*, pages 115–132, 2004.

[7] G.J. Holzmann. *The SPIN model checker: Primer and reference manual*. Addison-Wesley Professional, 2004.

[8] K. Homma, S. Izumi, K. Takahashi, and A. Togashi. Modeling, verification and testing of web applications using model checker. *IEICE Transactions on Information and Systems*, 94(5):989–999, 2011.

[9] N. Jovanovic, C. Kruegel, and E. Kirda. Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). *In IEEE Symposium on Security and privacy*, 2006.

[10] D.C. Kung, C.H. Liu, and P. Hsia. An object-oriented web test model for testing web applications. In *Conference on Quality Software. Proceedings. First Asia-Pacific*, pages 111–120. IEEE, 2000.

[11] H. Miao and H. Zeng. Model checking-based verification of web application. *Engineering Complex Computer Systems*, 2007.

[12] F. Ricca and P. Tonella. Analysis and testing of web applications. In *ICSE*, pages 25–34. IEEE Computer Society, 2001.

[13] A. Valmari. The state explosion problem. *Lectures on Petri Nets: advances in Petri Nets. Basic models*, page 429, 1998.

Timed Analysis of RFID Distance Bounding Protocols

Yu Lu[1] Alice Miller[2]

[1] School of Computing Science, University of Glasgow, UK, y.lu.3@research.gla.ac.uk
[2] School of Computing Science, University of Glasgow, UK, alice@dcs.gla.ac.uk

Abstract: Modelling real time is fundamental to reason about pervasive systems. The formal analysis of some time sensitive security protocols, such as distance bounding protocols, could lead to a more formal approach to time dependent properties formalisation and verification of pervasive systems.

1 Introduction

Pervasive systems often contain devices which must operate in very different environments and connect together in different ways, and still satisfy all the desired security properties. The rapid development of wireless technologies (such as RFID) has led to new application areas for pervasive systems with novel security requirements for the protocols employed. Unlike traditional security protocols concerning message secrecy or different types of authentication, these new protocols employed in new applications usually establish security properties coupled with the wireless network environment.

Physical location is used as a measurement of trust in wireless networks, RFID-based systems and vehicular communication that require secure localisation, time synchronisation, neighbour discovery, and neighbour verification. For such location services, it is crucial to securely estimate distance between two nodes in a wireless network and thus impede man-in-the-middle attacks. The main countermeasure against such attacks is the use of *Distance bounding (DB) protocols*. DB protocols are a class of identification protocols in which one "verifier" node in wireless networks measures an upper bound on its distance to another "prover" node in the network. Accordingly, the security of DB protocols is applicable to most pervasive computing applications.

So far, DB protocols have been extensively studied: a large number of protocols have been proposed and analysed in the past decade. Regardless of the different type of DB protocols, the distance bound is obtained from a rapid exchange of messages between the verifier and the prover in the fast bit phase. In this phase, the verifier sends a challenge to the prover, to which the prover responds after some processing time. The verifier measures the round-trip time between sending its challenge and receiving the responses from the prover, subtracts the prover's processing time, and based on the remaining time, computes the distance bound between the devices.

Typically, DB protocols are designed and analysed with respect to three different classes of attack scenarios:

- *Mafia fraud* attacks where the attacker $\mathcal{A}$ relays communication between a honest prover P and a honest verifier V in different sessions
- *Distance fraud* attacks where a malicious prover P claims to be closer to the verifier V than it actually is
- *Terrorist fraud* attacks where the attacker $\mathcal{A}$ gets limited active support from the prover P to deceive the verifier V

All attacks aim to make the verifier believe that the prover P is physically closer to the verifier V than it really it. Recently, a fourth type of real time attack on DB protocols, called *Distance hijacking* attacks, has been defined and analysed [4]. Although nowadays many proposed DB protocols are resistant to mafia fraud, verifying DB protocols using existing informal and formal frameworks still does not guarantee the absence of other attacks, e.g., the distance hijacking.

2 Related Work

The first DB protocol was proposed in [3] in 1993, but the first formal analysis of DB protocols was presented in 2007 ([9]). In [9], the authors not only proposes a new protocol for distance bounding that requires less message and cryptographic complexity, but also uses authentication and secrecy logics to analyse its security. Their logical framework is only based on qualitative analysis and does not provide any extended analysis of the timing properties. Since then, several quantitative frameworks for the verification of real time sensitive protocols have been proposed.

The constraint solver tool, which is a protocol security analyzer taking advantage of constraint programming techniques, was used to automatically analyse DB protocols in [8]. A natural limitation of their analysis is that it cannot tackle unbounded analysis since the constraint solver only considers bounded number of protocol processes. Meanwhile, a related approach to modelling and verifying physical properties (namely communication, location, and time) of DB protocols using HOL/Isabelle was presented in [10]. Being a verification effort, the two approaches in [8] and [10] differ in the classical way that model checking differs from theorem proving: the former tests for attacks while the latter proves their absence of.

It seems that since the introduction of the first RFID distance bounding protocol [7] in 2005, numerous DB protocols have been proposed, in an attempt to make them appropriate for the RFID systems. Unfortunately, many protocols in the literature address no rigorous cryptographic security models, nor the case of clear security proof. Also, they are commonly designed without any formal methods, which lead to inaccurate analyses. We consider that distance bounding for RFID systems is more difficult to achieve due to constrained resource of RFID tags.

During the last two years, there has been a recent surge in interest and research, in the arena of formal approaches to RFID-based distance bounding protocols. A new framework [6] was proposed, based on common game-based notion on cryptography, to analyse the security of the RFID DB protocols. Although this new approach addresses RFID authentication and can also be applied to general DB protocols, it still abstracts away from timed analysis. Another systematic method [2] aims to improve analysis and design of RFID DB protocols. Although the unified framework includes a thorough terminology about frauds, attackers, and provers, thus clarifying many misleading terms, the generic model only allows for the refinement of the security analysis, but not to verify security properties.

3 Our Approach

To the best of our knowledge, all the existing techniques for verifying security protocols specifically for pervasive systems abstract away from real time, focusing only on the sequencing of events. Although this has many advantages, it is a serious limitation for reasoning about RFID protocols for secure distance bounding, which rely on real time considerations. Furthermore, past efforts to analyse DB protocols have only been manual. Automated analysis would avoid the problems and distrust in manual analysis of protocols that have often been reported. Thus, we consider that automated approaches are critical since they are quite likely to find flaws that manual approaches cannot.

Our contributions will be threefold: (1) To give in-depth and rigorous analyses of how to formalise time dependent properties in security protocols using modelling languages such as applied pi calculus [1], (2) to define the time dependent security properties formally against attacks RFID distance bounding protocols could address. Finally, (3) we will extend existing formal verification techniques (such as model checking and process calculi), towards a automated verification of such protocols.

The most two popular approaches are based on automated methods, such as model checking, and interactive methods, such as theorem proving. In both scenarios, it is standard to formalise an intruder model based on the Dolev-Yao model [5], which identifies the intruder with the network. However, the conventional Dolev-Yao style analysis of security protocols is inappropriate to analyse DB protocols in our case. Analysis of RFID DB protocols involves examining whether it is possible to make a tag appear closer than it really is, to an honest reader. The problem is different and difficult compared to standard Dolev-Yao analysis of protocols that only consider whether an attacker can generate messages required to violate some security properties. Thus, we need to consider the timing required for generation and transmission as well.

Formal verification using automatic verifier ProVerif has been discussed in [8] as an extension of their analysis. In particular, it suggests adding four events in the DB protocols, two each for the verifier and prover, corresponding to sending and receiving the challenge and rapid response in the fast bit phase. The security property they formulate is a time-based trace equivalence that we plan to formalise in applied pi calculus as a starting point for our timed analysis.

4 Conclusion

The timed analysis of RFID distance bounding protocols will enable us to tackle the problem of modelling real-time aspects in timed process calculi and thus define and formally verify time dependent security properties. This will be essential to formally verify pervasive systems.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. POPL '01*, pages 104–115, 2001.

[2] G. Avoin, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011.

[3] S. Brands and D. Chaum. Distance-bounding protocols. *LNCS*, 765:344–359, 1994.

[4] C. Cremers, K. B. Rasmussen, and S. Capkun. Distance hijacking attacks on distance bounding protocols, 2011.

[5] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[6] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding RFID protocols. In *Proc. ISC'11*, pages 47–62, 2011.

[7] G. P. Hancke and M. G. Kuhn. An RFID distance bounding protocol. In *Proc. SECURECOMM '05*, pages 67–73, 2005.

[8] S. Malladi, B. Bruhadeshwar, and K. Kothapalli. Automatic analysis of distance bounding protocols. *CORR*, 2010.

[9] C. Meadows, R. Poovendran, D. Pavlovic, Chang L., and P. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. *Advances in Information Security*, 30(Part II):279–298, 2007.

[10] P. Schaller, B. Schmidt, D. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Proc. CSF '09*, pages 109–123, 2009.

Progress on Model Checking Robot Behaviour

Ryan Kirwan Alice Miller

[1] Department Of Computing Science, University Of Glasgow, `kirwanr@dcs.gla.ac.uk`
[2] Department Of Computing Science, University Of Glasgow, `alice@dcs.gla.ac.uk`

Abstract: We model systems that involve a learning robot which interacts with obstacles in a static environment. Our models are specified in Promela with a view to verifying them using SPIN [2]. Because of the complex nature of these systems our initial models have intractable state-spaces: we aren't able to model check them. Last year [4] we introduced an abstraction technique to greatly reduce the state-space of our models. One of the challenges we now face is to prove that our abstraction technique is correct i.e., it preserves the properties of concern.

1 Introduction

Traditionally, testing and simulation have been used to validate robot behaviour and to validate algorithms defining robot learning. Model checking [3] robot behaviour allows us to formally check properties and, potentially, to measure the performance of learning algorithms.

The robot behaviour captured in our models is as defined in [5]. Behaviour analysis is the study of how learning is affected by an environment and by the perception of the learner. Through simulations, the ability of a robot to successfully navigate its environment is used to assess the robot's sensor configuration and the learning algorithm it uses. These assessments are generated by averaging results from sets of simulated experiments. By applying model checking we can derive properties by a single search of the state-space (per property).

Our initial models are specified with a view to verification. We begin our modelling process by generating the *Explicit model* of the system. This is a highly detailed model with a state-space too large to verify. Our next step is to apply our abstraction technique [4] to the *Explicit model* to generate a tractable model, the *Relative model*. The current challenge is to prove the correctness of our abstraction technique.

2 Explicit model

The *Explicit model* uses polar coordinates to determine position within an environment. This allows us to calculate the turning angles of the robot more accurately than with a standard grid representation. All positions in the environment are stored as an angle and a distance relative to the centre of the environment.

The robot is modelled as the only moving object in an environment. As in the real systems it is composed of two distal and two proximal antennas. In our model we don't include the robot's motors or external wheels. The robot is simply modeled to represent its avoidance behaviour.

We use embedded *C* code, within the Promela specification, to calculate the precise location of the robot as it moves around an environment. The robot's angle from the centre and it's distance from the centre are used in the calculation. We also consider the angle that the robot is facing, relative to North.

The robot's learning is implemented with a simple calculation that involves incrementing a *learning factor* every time the robot collides into something. If the robot collides with something then it learns to respond more strongly to the type of signals that it received before the collision.

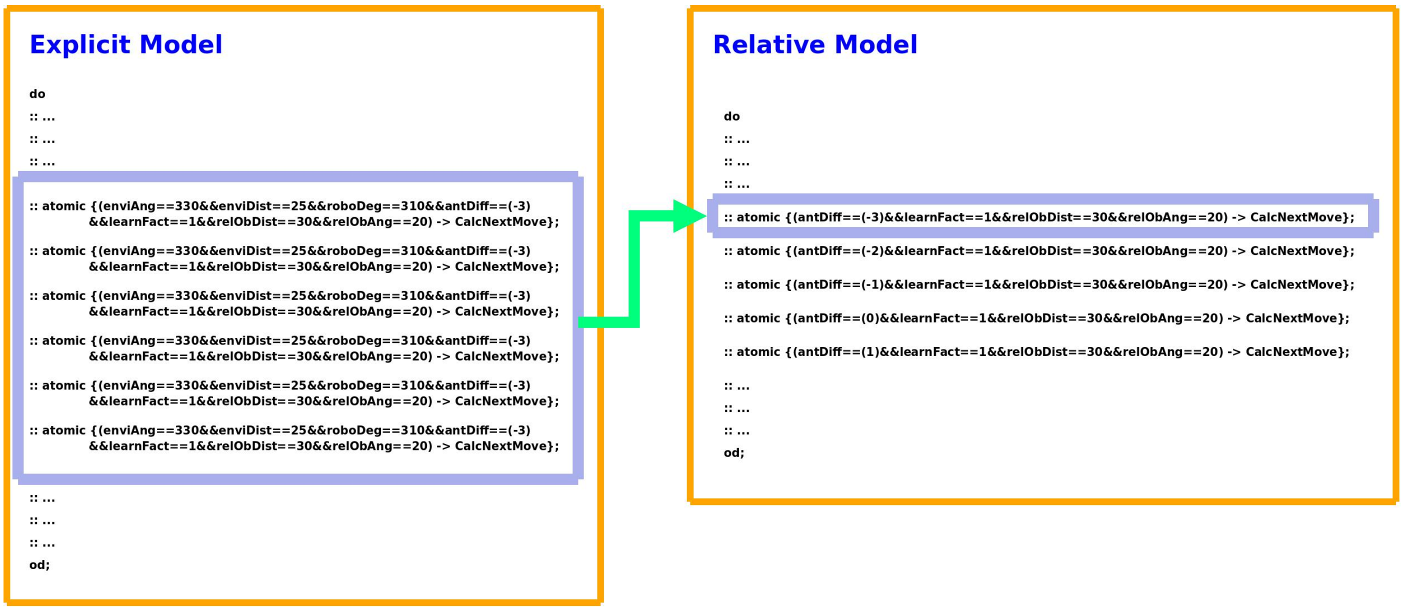
3 Relative model

The *Relative model* is our abstraction of the *Explicit model*. Unlike the *Explicit model* the robot is not represented as a set of coordinates and an orientation, but as the centre of a polar axis. We model an environment from the perspective of the robot, we will herein refer to the area that this perspective covers as the *cone of influence*.

The *cone of influence* has 80° of angle and is split into nine units of distance. Each unit of distance is the length of a proximal antenna (the smallest object in the system). The robot's learning is implemented in the same way as in the *Explicit model*. The resulting state-space is tractable, allowing us to verify LTL properties using model checking. E.g., the LTL formula "**[]** (ω <11) " checks that the robot's *learning factor* (ω) always remains less than 11.

4 Proving the equivalence of our models

One of the biggest challenges we face is proving the correctness of the *Relative model*. Our proof is based on one given in [6] where a similar proof is used for abstracted featured networks. We achieve this by converting our models to *Guarded Command Form* (GCF). We map sets of transitions in the *Explicit model* to transitions in the *Relative model*. An example of a Promela specification *GCF* is given in both sides of Figure 1. Once we have our models in *GCF* we map transitions, as shown in Figure 1.



```
do
:: ...
:: ...
:: ...
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(enviAng==330&&enviDist==25&&roboDeg==310&&antDiff==(-3)
          &&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: ...
:: ...
:: ...
od;
```

```
do
:: ...
:: ...
:: ...
:: atomic {(antDiff==(-3)&&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(antDiff==(-2)&&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(antDiff==(-1)&&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(antDiff==(0)&&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: atomic {(antDiff==(1)&&learnFact==1&&relObDist==30&&relObAng==20) -> CalcNextMove};
:: ...
:: ...
:: ...
od;
```

Figure 1: Equating guarded command line from Explicit to Relative model.

With our proof we hope to conclude that there is a simulation relation between the *Explicit model* ($\mathcal{M}$) and the *Relative model* ($\mathcal{M}'$). This relation implies that any property that holds for $\mathcal{M}'$ also holds for $\mathcal{M}$.

If $\mathcal{M}'$ simulates $\mathcal{M}$ then for every LTL formula ψ
$\mathcal{M}' \models \psi$ implies that $\mathcal{M} \models \psi$.

5 Future Work

We have begun to use PRISM [1] to create similar system models, but have not verified any quantitative properties yet. Principally, we want to develop a proven abstraction technique for systems that involve robot learning. One further aim is to develop a custom-made tool to automatically abstract and model check this type of system.

References

[1] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. *LNCS*, 3920/2006:441–444, 2006.

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

[3] Edmund M. C. Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, February 2000.

[4] R. Kirwan and M Alice. Abstraction for model checking robot behaviour. *Proceedings of ARW'11*, pages 1–2, April 2011.

[5] T. Kulvicius, C. Kolodziejski, T. M. Tamosiunaite, B. Porr, and F. Worgotter. Behavioral analysis of differential hebbian learning in closed-loop systems. *Biological cybernetics*, 2010.

[6] A. Miller, C. Calder, and A. Donaldson. A template-based approach for the generation of abstractable and reducible models of featured networks. *Computor Networks*, vol 51/2, February 2007.

SAT over an Abstract Domain

Daniel Kroening

University of Oxford

`http://www.cs.ox.ac.uk/people/daniel.kroening/`

Abstract: This is joint work with Vijay DSIlva, Leopold Haller, and Michael Tautschnig. We present a generalisation of the DPLL(T) framework to abstract domains. As an instance, we present a sound and complete analysis for determining the range of floating-point variables in embedded control software. Existing approaches to bounds analysis either use convex abstract domains and are efficient but imprecise, or use floating-point decision procedures, and are precise but do not scale. We present a new analysis that elevates the architecture of a modern SAT solver to operate over floating-point intervals. In experiments, our analyser is consistently more precise than a state-of-the-art static analyser and significantly outperforms floating-point decision procedures.

Deforestation + dependent types = automated induction

William Sonnex

University of Cambridge wcs23@cam.ac.uk

Abstract: We present an extension to Wadler's deforestation which can automatically simplify a twice reversed list (*rev (rev xs)*), to just the list itself (*xs*). We then show how the same deforestation algorithm can be used to construct a proof by induction of the validity of this simplification as a dependently typed program.

1 Deforestation

Walder's deforestation[2] removes intermediary trees from functional programs, aimed at optimizing runtime speed. In Figure 1 we give an example where $T[\![E]\!]$ is the *deforestation* of E. At (1.1) we define a new arity-2 function *rev2 xs n* to be the result of deforesting *rev (xs ++ [n])*. We apply β-reduction within our deforestation to get to (1.3). To get to (1.4) we then use a rule from [2], whereby a case-analysed case-analysis is pushed to the top and the original topmost case-analysis is pushed into each of its branches; deforestation can then move into these branches as we have a case-analysed variable topmost (*xs*). β-reduction gets us to (1.5). Finally (1.6) is the result of applying our original definition in (1.1) for *rev2*, but with *cs* for *xs*, and finishing deforestation. We now have a recursive definition for *rev2* which only traverses our input list once, as opposed to *rev (xs ++ [n])* which traverses it twice.

$$
\begin{array}{l}
\text{as ++ bs} = \textbf{case}\text{ as }\textbf{of}\ \{\ [] \to \text{ys};\ (\text{c:cs}) \to \text{c:(cs ++ bs)}\ \}\\
\text{rev ds} = \textbf{case}\text{ ds }\textbf{of}\ \{\ [] \to [];\ (\text{e:es}) \to (\text{rev es}) \text{ ++ [e]}\ \}
\end{array}
$$

$$
\begin{array}{lll}
\text{rev2 xs n} & \triangleq & T[\![\text{rev (xs ++ [n])}]\!] \hfill (1.1)\\
& =_\beta & T[\![\textbf{case}\text{ (xs ++ [n]) }\textbf{of}\ \{\\
& & \quad [] \to [];\ (\text{e:es}) \to (\text{rev es}) \text{ ++ [e] }\}]\!] \hfill (1.2)\\
& =_\beta & T[\![\textbf{case}\ (\textbf{case}\text{ xs }\textbf{of}\ \{\\
& & \quad [] \to [\text{n}];\ (\text{c:cs}) \to \text{c:(cs ++ [n])}\ \})\ \textbf{of}\\
& & \quad [] \to [];\ (\text{e:es}) \to (\text{rev es}) \text{ ++ [e] }\}]\!] \hfill (1.3)\\
& =_W & \textbf{case}\text{ xs }\textbf{of}\ \{\\
& & \quad [] \to T[\![\textbf{case}\text{ [n] }\textbf{of}\ \{\\
& & \quad\quad [] \to [];\ (\text{e:es}) \to (\text{rev es}) \text{ ++ [e] }\}]\!];\\
& & \quad (\text{c:cs}) \to T[\![\textbf{case}\text{ c:(cs ++ [n]) }\textbf{of}\ \{\\
& & \quad\quad [] \to [];\ (\text{e:es}) \to (\text{rev es}) \text{ ++ [e] }\}]\!] \hfill (1.4)\\
& =_\beta & \textbf{case}\text{ xs }\textbf{of}\ \{\ [] \to T[\![[\text{n}]]\!];\\
& & \quad (\text{c:cs}) \to T[\![(\text{rev (cs ++ [n])) ++ [c]}]\!]\ \} \hfill (1.5)\\
& = & \textbf{case}\text{ xs }\textbf{of}\ \{\ [] \to [\text{n}];\\
& & \quad (\text{c:cs}) \to (\text{rev2 cs}) \text{ ++ [c] }\} \hfill (1.6)
\end{array}
$$

Figure 1: Wadler's deforestation

2 Deforestation within contexts

Our first extension comes from the fact that *n:(rev xs)* is a simpler form of *rev (xs ++ [n])* than *rev2 xs n*. Ordinary deforestation cannot simplify a term to head normal form. The first step is to discover this static context which the recursive function sits inside (e.g. *(n:)*). For this we take a dynamic approach and enumerate inputs to our original term to find a common context in the outputs, e.g. taking $E \triangleq$ *rev (xs ++ [n])*: E[xs := []] $=_\beta$ [n], E[xs := [a]] $=_\beta$ [n, a], E[xs := [a, b]] $=_\beta$ [n, b, a] . So the common context can be recognised as *(n:)*. In Figure 2 we use this information to assume the existence of a function *rev3 xs n*, such that *rev (xs ++ [n])* is *n:(rev3 xs n)* (2.1). We now perform deforestation within the context *(n:)*, where $\langle C\rangle[\![E]\!]$ is the deforestation of E in context C. This acts identically to our previous deforestation example up to (2.3), first expanding then rewriting (also applying ++ to move *(n:)* topmost). The difference with ordinary deforestation is that we cannot finish deforesting a branch until its topmost term is within our context, but since *[n]* and *n:((rev3 xs n) ++ [c])* are, we remove this outer context and finish. By removing the unused parameter *n* we now have a definition for *rev3* which is α-equivalent to our original definition for *rev*.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{n:(rev3 xs n)}}\\
\triangleq & \langle\text{n:}\rangle[\![\text{rev (xs ++ [n])}]\!] \hfill (2.1)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{\ [] \to \langle\text{n:}\rangle[\![[\text{n}]]\!];\\
& \quad (\text{c:cs}) \to \langle\text{n:}\rangle[\![(\text{rev (cs ++ [n])) ++ [c]}]\!]\ \} \hfill (2.2)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{\ [] \to \langle\text{n:}\rangle[\![[\text{n}]]\!];\\
& \quad (\text{c:cs}) \to \langle\text{n:}\rangle[\![\text{n:((rev3 xs n) ++ [c])}]\!]\ \} \hfill (2.3)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{\ [] \to [];\\
& \quad (\text{c:cs}) \to (\text{rev3 cs n}) \text{ ++ [c] }\} \hfill (2.4)
\end{array}
$$

Figure 2: Deforestation inside a context

Using this simplification we can deforest *rev (rev xs)* into *id_rec xs* (a recursive identity function on lists), shown in Figure 3. Normal deforestation occurs up to (3.2) then we get to (3.3) by applying our simplification from Figure 2 and removing the unnecessary *n* parameter. We then perform our substitution (modulo α-equivalence) from the definition in (3.1) to get to (3.4). Since deforestation cannot remove the final tree of *id_rec* we have an additional technique which removes recursive identity functions.

3 Inductive theorem proving as deforestation

Having discovered that *rev (xs ++ [n])* $\equiv$ *n:(rev xs)* and *rev (rev xs)* $\equiv$ *xs* we would now like to prove it. We found that the generation of such a proof in dependent type theory can be done using another new form of deforestation, which we call P-deforestation. Figure 4 is an example.

id_rec xs

$$
\begin{array}{lll}
\triangleq & T[\![\text{rev (rev xs)}]\!] & (3.1)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow T[\![[\,]]\!]; & \\
 & \quad \text{(c:cs)} \rightarrow T[\![\text{rev ((rev cs) ++ [c])}]\!]\ \} & (3.2)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow [\,]; & \\
 & \quad \text{(c:cs)} \rightarrow T[\![\text{c:(rev3 (rev cs))}]\!]\ \} & (3.3)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow [\,]; & \\
 & \quad \text{(c:cs)} \rightarrow \text{c:(id\_rec cs)}\ \} & (3.4)
\end{array}
$$

Figure 3: Deforesting *rev (rev xs)*

refl :: a ≡ a

trans :: a ≡ b → b ≡ c → a ≡ c

cong :: f → a ≡ b → f a ≡ f b

(revapp xs n) :: rev (xs ++ [n]) ≡ n:(rev xs)

$$
\begin{array}{lll}
\triangleq & P[\![\text{rev (xs ++ [n])} \equiv \text{n:(rev xs)}]\!] & (4.1)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow P[\![\text{[n]} \equiv \text{[n]}]\!]; & \\
 & \quad \text{(c:cs)} \rightarrow & \\
 & \qquad P[\![\text{rev (cs ++ [n]) ++ [c]} \equiv \text{n:(rev cs ++ [c])}]\!]\} & (4.2)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow P[\![\text{[n]} \equiv \text{[n]}]\!]; & \\
 & \quad \text{(c:cs)} \rightarrow \text{trans (cong (++ [c]) (revapp cs n))} & \\
 & \qquad P[\![\text{n:(rev cs ++ [c])} \equiv \text{n:(rev cs ++ [c])}]\!]\} & (4.3)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow \text{refl}; & \\
 & \quad \text{(c:cs)} \rightarrow \text{trans (cong (++ [c]) (revapp cs n)) refl} & (4.4)
\end{array}
$$

Figure 4: Proving rev (xs ++ [n]) ≡ n:(rev xs)

(revrev xs) :: rev (rev xs) ≡ xs

$$
\begin{array}{lll}
\triangleq & P[\![\text{rev (rev xs)} \equiv \text{xs}]\!] & (5.1)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow P[\![[\,] \equiv [\,]]\!]; & \\
 & \quad \text{(c:cs)} \rightarrow P[\![\text{rev (rev cs ++ [c])} \equiv \text{c:cs}]\!]\} & (5.2)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow \text{refl}; & \\
 & \quad \text{(c:cs)} \rightarrow \text{trans (revapp (rev cs) c)} & \\
 & \qquad (P[\![\text{c:(rev (rev cs))} \equiv \text{c:cs}]\!]\}) & (5.3)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow \text{refl}; & \\
 & \quad \text{(c:cs)} \rightarrow \text{trans (revapp (rev cs) c)} & \\
 & \qquad \text{(trans (revrev cs)}\ (P[\![\text{c:cs} \equiv \text{c:cs}]\!])) & (5.4)\\
= & \textbf{case}\text{ xs }\textbf{of}\ \{ & \\
 & \quad [\,] \rightarrow \text{refl}; & \\
 & \quad \text{(c:cs)} \rightarrow \text{trans (revapp (rev cs) c)} & \\
 & \qquad \text{(trans (revrev cs) refl)} & (5.5)
\end{array}
$$

Figure 5: Proving rev (rev xs) ≡ xs

At (4.1) we define a new function *revapp xs n*, whose type is our property, to be equal to the P-deforestation of the property. Getting to (4.2) uses the expansion from (1.1) to (1.5), we then float the case-analysis outside of the P-deforestation, as it is over a variable (*xs*). Since *rev (cs ++ [n])* from (4.2) is an instance of the LHS of our originally defined equality we can rewrite it to *n:(rev cs)* in (4.3), this is equivalent to applying the inductive hypothesis in a "regular" induction proof and is manifested in the recursive call to *revapp cs n*. Here *trans* and *cong* are machinery to indicate how this rewrite is applied and will be familiar to Agda users. Finally in (4.4) we translate any $P[\![E \equiv E]\!]$ into just *refl*(exivity of equality). This final definition for *revapp* in (4.4) gives us a recursive, dependently typed function representing our proof.

Using this process we then prove *rev (rev xs)* ≡ *xs* in Figure 5. In getting from (5.2) to (5.3) the algorithm has detected the available simplification from Figure 2 (i.e. *rev (rev cs ++ [c])* to *c:(rev (rev cs))*) and applied it (using the proof-term *revapp*). It then detects the potential to rewrite *c:(rev (rev cs))* to *c:cs* by calling *revrev cs* at (5.4) and the proof finishes with *refl* at (5.5).

4 Controlling deforestation

One difficulty with this technique is controlling which function definitions are expanded when, to ensure termination. Wadler's original work relied on all defined functions being of a restricted shape in order to ensure this. Having found this correspondence with inductive theorem proving we have instead taken our earlier work [1] on controlling this process using *critical pairs* and adapted it to our deforestation algorithm. We found that it worked perfectly with not only our P-deforestation but our $\langle C\rangle$ version and Wadler's T version. Unfortunately we have no proof of termination for this technique, but we are optimistic this can be produced with further work.

5 Conclusion

We have shown a fully automated technique which will transform the term *rev (rev xs)* to *xs* and how the same technique can also be used to generate a proof term verifying this transformation. So far we have implemented a working simplifier (which can do the *rev (rev xs)* simplification) but not the inductive theorem prover. Another interesting transformation it does is *length (rev xs)* to *length xs*.

References

[1] William Sonnex, Sophia Drossopoulou, and Susan Eisenbach. Zeno: An automated prover for properties of recursive data structures. In *TACAS*, 2012.

[2] Philip Wadler. Deforestation: transforming programs to eliminate trees. *Theoretical Computer Science*, 73(2):231 – 248, 1990.

Using Algebra to Understand Search Spaces

Martin Brain[1]

Department of Computer Science, University of Oxford, Oxford, OX1 3QD
martin.brain@cs.ox.ac.uk

Abstract: Existing formal approaches to understanding the performance of SAT, SMT, CSP and ASP solvers are language or algorithm specific. As such, they tend to be of limited applicability and tied to the detail of language syntax. This paper proposes an algebraic formalism for understanding combinatorial search. Results include a characterisation of deterministic inference and a language independent metric of problem difficulty.

1 Introduction

Over the past two decades tools such as SAT, SMT, CSP and ASP solvers have become widely used in academic and industrial fields including AI, planning and verification. Improvements to the performance of these solvers strengthen a wide range of applications and enable new areas that were previously impractical. Progress has been rapid and impressive but is practically focused; algorithms are evaluated by timed benchmarks of implementation and there is little explanation of *why* certain techniques work better than others. There is a need for a formalism that enables one to discuss the *process* of reasoning and searching, as opposed to the *result* of reasoning.

This paper describes an ongoing program of work aimed at developing a suitable theoretical framework based on abstract algebra. The aims of developing a new formalism and the advantages of using algebra are described in Section 2. The key mathematical structure is described in Section 3 with Section 4 describing inference and reasoning used and Section 5 giving an algebraic model of difficulty. Finally, Section 6 gives some of the future directions for this work.

2 Motivation

Most fields that develop combinatorial search or model generation tools have some language for describing their algorithms and executions. For example, proof complexity in SAT [2], abstract solver frameworks in SAT and ASP [5] and the numerous uses of trees in CSP [3]. Thus it is important to understand what advantages the new approach gives:

1. It is representation-independent and not tied to any particular style of language semantics. This makes results much more general and allows for unification between different fields.

2. It allows formalisation of both search spaces and the algorithms used to traverse them. This can be used to show how the behaviour of solver algorithms is influenced by the properties of the search space and to provide implementation and algorithm independent metrics.

3. It lays the foundations for discussions of structure and its role in making combinatorial search tractable.

To achieve this, there are two key innovations:

1. Rather than starting from a language specific structure, an abstract algebra is used to represent a search space. This allows development analogous to that of group or ring theory and the re-use of existing results from universal algebra.

2. Instead of just formalising the trace of a solver, the whole search space is formalised with execution traces appearing as substructures.

3 I-Spaces

Search spaces are modelled using a mathematical structure called an I-Space:

Definition 1 *Let* $\mathbb{I} = (L, \leqslant, \Phi, S)$. $\mathbb{I}$ *is an an* I-Space *under the following conditions:*

$$\begin{aligned} ISpace(\mathbb{I}) \quad \Leftrightarrow \quad & latticeOrder(L, \leqslant) \wedge \\ & (\Phi \in L) \wedge (\forall I \in L \,.\, I \leqslant \Phi) \wedge \\ & (S \subseteq L \setminus \{\Phi\}) \wedge \\ & (\forall I_s \in S \,.\, {}^{\uparrow}I_s \setminus \{\Phi\} \subseteq S) \end{aligned}$$

For example, the propositional logic formulae:

$$(a \vee \neg b) \wedge (a \vee b) \wedge b$$

generates the I-Space shown in Figure 1. The base set (commonly denoted L) is the set of all partial interpretations of the atoms a and b (written as disjoint pairs of sets) plus an extra point, Φ, at the top. The additional point means that when the partial interpretations are ordered by inclusion they form a lattice order. The final part of the definition is a set of 'solutions'; the things that are sought in the search space. In the case of the example, this is the single interpretation with both a and b true.

4 Homomorphisms and Inference Functions

One of the first steps when exploring a mathematical structure is to define the concept of a homomorphism. In the case of I-Spaces, these are maps that preserve the ordering and conserve the solutions and solubility of the space:

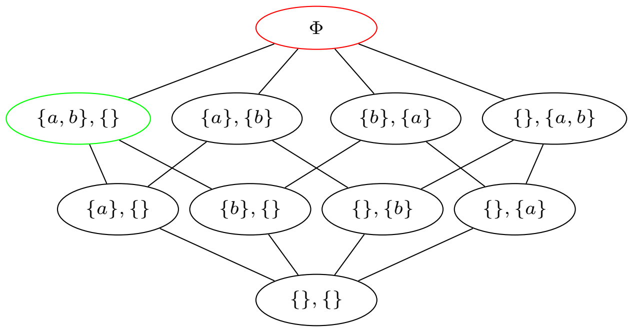


Figure 1: Partial interpretations for the set $\{a, b\}$ augmented with Φ

Definition 2 *Given I-Spaces $\mathbb{I}_1 = (L_1, \leqslant, \Phi_1, S_1)$ and $\mathbb{I}_2 = (L_2, \preccurlyeq, \Phi_2, S_2)$, a function $h : L_1 \to L_2$ is a* lax homomorphism *under the following conditions:*

$$\begin{aligned}&laxHomomorphism(h, \mathbb{I}_1, \mathbb{I}_2) \Leftrightarrow\\&\quad(\forall I_1, I_2 \in L_1 \,.\, I_1 \leqslant I_2 \Rightarrow h(I_1) \preccurlyeq h(I_2))\wedge\\&\quad h(\Phi_1) = \Phi_2 \wedge h(S_1) \subseteq S_2\end{aligned}$$

Homomorphisms form the basis of the model of inference. The idea of inference or reasoning used is very general; it is *a deterministic method of adding information that necessarily holds if the point in question is part of a solution*. It is important to note that inference functions are intended to capture the *effect* of reasoning, not the *mechanism*.

Definition 3 *Given an I-Space $\mathbb{I} = (L, \leqslant, \Phi, S)$ a function $e : L \to L$ is an* inference function *if it meets the following criteria:*

$$\begin{aligned}&inferenceFunction(e, \mathbb{I}) \Leftrightarrow\\&\quad laxHomomorphism(e, \mathbb{I}, \mathbb{I})\wedge\\&\quad closureOperator(e, (L, \leqslant))\end{aligned}$$

The use of closure operators to model reasoning is not novel [1, 6]. However when combined with the explicit model of search space, it has a number of useful properties. For example, the space of inference functions (on a given I-Space) form a lattice when ordered pointwise. The corresponding join operator is:

$$(e_1 \cup e_2)(I) = \bigcup_{n \in \mathbb{Z}} (e_1 \circ e_2)^n(I)$$

which can be found directly implemented in ASP, CSP and SMT solvers as well as many hybrid reasoning systems. Furthermore, the space of inference functions can be shown to form an I-Space. Homomorphisms on this space seem a theoretical construct but correspond to 'meta-heuristics' such as lookahead.

5 Models of Difficulty

A key concept when solving search problems is whether or not the current partial information is part of a solution. Determining this computationally is often as hard (or harder) than solving the problem. However it can be defined mathematically:

Definition 4 *Given an I-Space $\mathbb{I} = (L, \leqslant, \Phi, S)$, an element $I \in L$ is* green *or* red *in $\mathbb{I}$ under the following conditions:*

$$\begin{aligned}green(I, \mathbb{I}) &\Leftrightarrow {}^{\uparrow}I \cap S \neq \emptyset\\red(I, \mathbb{I}) &\Leftrightarrow \neg green(I, \mathbb{I})\end{aligned}$$

Defining the "solubility" of a point as a colour allows a representation and algorithm dependent definition of difficulty:

Definition 5 *Given an I-Space $\mathbb{I} = (L, \leqslant, \Phi, S)$, a* path *is a totally ordered subset of L and the* difficulty *of $\mathbb{I}$ is the length of the longest red path.*

This gives a bound on the complexity of performing DPLL-like depth first search. Furthermore it is equal to proof width (when the I-Space corresponds to a resolution proof) and is bounded by induced width (when the I-Space corresponds to a Constraint Satisfaction problem).

6 Conclusion

I-Spaces give an algebraic account of combinatorial search, including inference and difficulty. Although still under exploration, they have the potential to unify the study of search across a number of areas, provide techniques for implementation and algorithm independent performance metrics and to serve as a foundation for understanding of the role of problem representation and structure in search.

References

[1] Krzysztof R. Apt. The role of commutativity in constraint propagation algorithms. *ACM Trans. Program. Lang. Syst.*, 22:1002–1036, November 2000.

[2] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artifical Intelligence and Applications*. IOS Press, Nieuwe Hemweg 6B, 1013 BG Amsterdam, Netherlands, 1st edition, February 2009.

[3] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

[4] Steven R. Givant and Ralph N. McKenzie, editors. *Alfred Tarski, Collected Papers*, volume 4. Birkhäuser, 1st edition, 1986.

[5] Yuliya Lierler. Abstract answer set solvers. In *Proceedings of the 24th International Conference on Logic Programming*, ICLP '08, pages 377–391, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] Alfred Tarski. Remarques sur les notions fondamentales de la méthodologie des mathématiques. *Annales de la Société Polonaise de Mathématique*, 7:270–272, 1928. Reprinted in [4].

Automated Reasoning in Higher-Order Algebra

Alasdair Armstrong
a.armstrong@dcs.shef.ac.uk

Georg Struth
g.struth@dcs.shef.ac.uk

Department of Computer Science, University of Sheffield

Abstract: We report on the development of a repository for algebraic methods in Isabelle/HOL. It is based on Isabelle's integrated ATP systems and SMT solvers. We are currently implementing higher-order properties such as directed sets, continuity, fixpoint calculi, Galois connections and set-theoretic models. This often combines algebraic reasoning with reasoning about numbers, sets and inductive data types.

1 Overview

Automated reasoning has recently made considerable impact on the area of algebraic methods. Calculational proofs at textbook level can often be fully automated; more difficult theorems can usually be mechanised at the granularity of paper and pencil proofs. A very suitable proof environment is Isabelle/HOL with its recent integration of ATP systems and SMT solvers through the Sledgehammer tool (c.f. [3]), which complements the sheer power of automated reasoning with higher-order facilities for proof management and theory engineering. Isabelle's type classes and locales support the design of theory hierarchies, the propagation of theorems across them and the formal connection of abstract algebras with concrete models. Isabelle's support for higher-order logic yields (co)inductive proofs and data types as well as mechanisms for abstraction and instantiation.

We are currently developing a large repository for algebraic methods in Isabelle that is strongly based on ATP[1]. Its purpose is to serve as a reference of the state-of-the-art in that area, to support the specification and verification of computing systems, and to help the working mathematician in exploring new variants. A main focus is on Tarski's relation algebras and variants of Kleene algebras.

The repository so far contains about 2000 facts, mostly of calculational nature. The next step is to explore higher-order features by ATP: to extract and generalise common features among algebras; to link existing first-order and higher-order variants; to provide simpler, more abstract and more generic proof mechanisms. Here we use variants of Kleene algebras to illustrate these features.

2 Kleene Algebras

A *dioid* is a structure $(D, +, \cdot, 0, 1)$, where $(D, +, 0)$ is a semilattice with least element 0, $(D, \cdot, 1)$ is a monoid, $\cdot$ distributes over $+$ from the left and right, and 0 is a left and right annihilator ($0 \cdot x = 0 = x \cdot 0$). A *Kleene algebra* [6] is a dioid expanded by the unary operation * that satisfies the unfold axioms $1 + xx^\star = x^*$ and $1 + x^\star x = x^*$ as well as the induction laws $z + xy \leq y \Rightarrow x^*z \leq y$ and $z + yx \leq y \Rightarrow zx^* \leq y$. Here, $\leq$ is the usual semilattice order defined by $x \leq y \Leftrightarrow x + y = y$.

[1] http://staffwww.dcs.shef.ac.uk/people/G.Struth/isa/

Suppose the elements of a Kleene algebra represent the action of some system. Then $x + y$ models the nondeterministic choice between x and y, xy their sequential composition, 1 the ineffective action (skip) and 0 the abortive one. x^* models a finite iteration of x, essentially as a least fixpoint of $\lambda y.1 + xy$ (and $\lambda y.1 + yx$). Kleene algebras have many interesting models, e.g., regular languages (see below) and binary relations under union, relational composition and the reflexive-transitive closure operation. Our repository contains a large number of facts about Kleene algebras, many variants and the most important models.

3 Higher-Order Variants

We are currently implementing more expressive variants called *regular algebras* [4, 2]. They are usually based on quantales, i.e. complete lattices that satisfy infinite distributivity laws with respect to suprema. Backhouse, in particular, requires the equivalent condition that the maps $\lambda y.xy$ and $\lambda y.yx$ are lower adjoints of a Galois connection. Formally, given two posets $(A, \leq_A)$ and $(B, \leq_B)$, a pair of functions $f : A \to B$ and $g : B \to A$ is a Galois connection between A and B iff $f(x) \leq_B y \Leftrightarrow x \leq_A g(y)$ holds for all $x \in A$ and $y \in B$. The function f is called the *lower adjoint* and g the *upper adjoint* of the Galois connection.

In this higher-order setting, the existence of the Kleene star as the least fixpoint of the isotone map $\lambda y.1 + xy$ is guaranteed by the Knaster-Tarski theorem. More generally, quantales support a fixpoint calculus for isotone functions over the underlying complete lattice.

Alternatively, a $*$-*continuous Kleene algebra* [6] is a dioid expanded by the Kleene star that satisfies $xy^\star z = \sup_n xy^n z$ where sup is defined with respect to the semilattice order. We have shown that every $*$-continuous Kleene algebra is a Kleene algebra. This requires induction, but the base case and induction step are essentially automatic.

A powerful tool in this context are *fixpoint fusion* laws (c.f. [2]). Roughly, if f is a lower adjoint in a Galois connection on a complete lattice and if g and h are isotone functions on that lattice, then $f \circ g = h \circ f \Rightarrow f(\mu g) = \mu h$, where μg and μh denote the least fixpoints of g and h. A dual fusion law holds for greatest fixpoints.

We have implemented regular algebras, $*$-continuous Kleene algebras and Galois connections in Isabelle. The Knaster-Tarski and fixpoint fusion theorems are work in progress; they reconstruct and extend previous approaches in Isabelle by ATP.

4 Examples

We now discuss some examples that highlight the applicability of ATP in higher-order algebra. We focus on Galois connections, which are are interesting because they yield theorems for free, and on fixpoint fusion.

Action Algebras These are first-order variants of regular algebras in which $\lambda y.xy$ and $\lambda y.yx$ are lower adjoints of the residuals $\lambda y.x \to y$ and $\lambda y.y \leftarrow x$. By general properties of Galois connections, residuation can be equationally axiomatised; more interestingly the Kleene star can be equationally axiomatised as well [7]. Having formally linked the Galois connections of action algebra with its general definition in Isabelle, several laws about residuals can be automatically instantiated. The proof that every action algebra is a Kleene algebra is fully automatic. Using Isabelle's locale mechanism, all theorems about Kleene algebras are then automatically inherited by action algebras.

Modal Kleene Algebras Dioids and Kleene algebras can be expanded by domain and range operations that abstractly model those on binary relations, e.g., $d(R) = \{(x,x) : \exists y.(x,y) \in R\}$ [5]. Based on these, abstract image and preimage operators can be defined. They give rise to forward and backward box and diamond operators, e.g. $|x\rangle y = d(xd(y))$ and $\langle x|y = r(r(x)y)$. Boxes and diamonds are adjoints in Galois connections, which automatically yield modal theorems in Isabelle. Next, a divergence (or nontermination) operator can be defined that models states in a system from which infinite sequences of actions may start, as a greatest fixpoint using $\nabla(x) = |x\rangle\nabla(x)$ and $d(y) \leq |x\rangle d(y) + d(z) \Rightarrow d(y) \leq \nabla(x) + |x\rangle d(z)$. Greatest fixpoint fusion shows that the second axiom is equivalent to $d(y) \leq |x\rangle d(y) \Rightarrow d(y) \leq \nabla(x)$ in Isabelle.

Solving Systems of Regular Equations The induction axioms of Kleene algebra can be strengthened, e.g., to $z + xy = y \Leftrightarrow x^*z = y$, to solve the regular equation $z + xy = y$ in y. An application is the extraction of regular expressions from automata. This, however, requires side conditions. We call x *deflationary* if $\forall y.(y \leq xy \Rightarrow y = 0)$ and *strongly deflationary* if $\forall y,z.(y \leq xy + z \Rightarrow y \leq x^*z)$. It is easy to show by ATP that strong deflationarity implies deflationarity. The converse direction can be proved in regular algebra by fixpoint fusion in Isabelle.

Language Kleene Algebras and Arden's Rule Isabelle supports a seamless transition between algebras and their models; e.g. Kleene algebra and the language model. In Isabelle, words are represented as lists; languages as sets of lists. The product of two languages X and Y is defined as $XY = \{xy : x \in X, y \in Y\}$ and $X^* = \sup_{i \geq 0} X^i$. Since ATP is rather fragile in this setting, we use rules such as $z \in XY \Leftrightarrow \exists x,y.z = xy \wedge x \in X \wedge y \in y$ and $x \in X^* \Leftrightarrow \exists i.x \in X^i$ to reduce to first-order reasoning. Isabelle's simplifier can then be called with these rules before ATP. This approach makes it straightforward to prove that regular languages form $*$-continuous Kleene algebras in Isabelle. All algebraic facts about Kleene algebras are then available for regular languages. This is very convenient, e.g., for proving facts such as Arden's rule [1], which states that if a language X does not contain the empty word, then $Z + YX = Y$ implies $ZX^* = Y$. We can show by induction that $y \leq yx + z \Rightarrow yx^{i+1} + zx^*$ holds for all i in Kleene algebra. Both the base case and the induction step are by ATP. In the language model we can then show that the term YX^{i+1} vanishes if X does not contain the empty word and i is sufficiently large. The proof is again inductive and essentially automatic. Arden's rule then follows from this and the second induction axiom of Kleene algebra. Apart from one single descent to the language model, the entire proof is fully algebraic.

5 Conclusion

These examples support our claim that Isabelle's integration of ATP systems and SMT solvers supports algebraic reasoning beyond pure first-order logic. We believe that this yields a new and particularly simple style of interactive theorem proving in which trivial and routine proof steps can by and large be discharged automatically. With regular and relation algebra, this allows the seamless integration of higher-order, pointfree algebraic and pointwise model-based reasoning with a high degree of flexibility and automation. In the future we plan to apply our repository in the development and verification of sequential and concurrent programs.

References

[1] D. N. Arden. Delayed-logic and finite-state machines. *FOCS 1961*, 0:133–151, 1961.

[2] R. C. Backhouse. Regular algebra applied to language problems. *J. Log. Algebr. Program.*, 66(2):71–111, 2006.

[3] J. C. Blanchette, L. Bulwahn, and T. Nipkow. Automatic proof and disproof in Isabelle/HOL. In C. Tinelli and V. Sofronie-Stokkermans, editors, *FroCos 2011*, volume 6989 of *LNAI*, pages 12–27. Springer, 2011.

[4] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[5] J. Desharnais and G. Struth. Internal axioms for domain semirings. *Sci. Comput. Program.*, 76(3):181–203, 2011.

[6] D. Kozen. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Information and Computation*, 110(2):366–390, 1994.

[7] V. Pratt. Action Logic and Pure Induction. In J. van Eijck, editor, *JELIA '90*, volume 478 of *LNCS*, pages 97–120. Springer, 1990.

Generation of Large Size Quasigroup Structures Using Algebraic Constraints

Quratul-ain Mahesar Volker Sorge

School of Computer Science,
The University of Birmingham
Birmingham, UK
Q.Mahesar|V.Sorge@cs.bham.ac.uk

Abstract: The generation of finite algebraic structures with particular properties is useful when they are needed in the context of concrete application. For algebraic structures more general than groups this is a very challenging computational problem. We focus on the generation of quasigroups using a combination of symbolic algebra computation and automated reasoning techniques. We present two algebraic methods for restricting the search space for a constraint solver: Firstly we compute and evolve generating system for quasigroups and secondly we filter randomly pre-computed elements by automated theorem prover.

1 Introduction

Quasigroups are non-associative algebraic structures whose operation has to satisfy only a single axiom, the Latin square property. There exists a very large number of different finite quasigroups 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. However, the lack of structure makes them difficult to handle algebraically, in particular to enumerate or to classify. We have developed methods to automatically generate quasigroups of large size by bootstrapping structural properties of smaller size quasigroups, instead of exhaustive search. We use bespoke symbolic algebra computations for the construction of generating systems that allow an easier computation of large size structures. This construction employs automated theorem proving to ensure a goal-directed generation of quasigroup-structures with particular 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.

The concept of generating systems was first introduced in [6], where classification theorems in quasigroup theory were generated incorporating a set of diverse reasoning techniques. We have further exploited these techniques and in particular the concept of generating system introduced in that work for the goal directed construction of quasigroups. In our work, we generate quasigroups with particular properties for small sizes using the model generator Mace4 [4]. The computed generating systems are then evolved for larger size quasigroup structures using bespoke symbolic algebra manipulations. We then give an alternative method that aims at using little structural knowledge by simply generating random elements for a Cayley table. But in order to rule out immediate failure due to property violation we employ a two stage filter process by filtering single elements via symbolic computation techniques and sets of elements via an automated theorem prover. Both techniques as well as their combination are encoded as constraint satisfaction problems to be solved by the constraint solver Minion [2]. Our experimental results demonstrate that our proposed approaches and their combination outperform existing techniques, for difficult, non-equational properties, enabling the generation of quasigroups that were previously unattainable.

2 Generating Systems Approach

The concept of generating systems for quasigroups was introduced in [6] 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 by a relation or factorisation.*

The generating systems computed from small size quasigroups are further evolved to generating systems sufficient for larger structures. This can essentially be achieved in two different ways: (a) by adding a new element as a generator, or (b) by expressing the new element as a relation in the existing generators. The newly added element is verified by our symbolic algebra system, that it does not violate the desired property of the quasigroup including the Latin square property. More formally we define the expansion of a generating system as follows:

Let $(Q, *)$ be a quasigroup of size n, i.e., $Q = \{0, \ldots, n-1\}$, with generating system $G = \langle S|R \rangle$. Then we can obtain a generating system G' by either one of the two steps:

(i) $G = \langle S \cup \{n\} | R \cup \{n = n\} \rangle$

(ii) $G = \langle S | R \cup \{n = w(s_1, \ldots, s_k)\} \rangle$, where $s_1, \ldots, s_k \in S$.

The resulting generating systems for the desired size and property of quasigroup are then verified by the automated theorem prover Prover9 [3], such that they do not violate the desired quasigroup property.

3 Element Filtering Approach

In the element filtering approach, for a quasigroup Q, we randomly generate triples that are added to a set of the form $S = \{(r, c, e) \| r, c, e \in Q\}$. Every time an element is added, we use a symbolic verification function to check that all the elements in the set are unique and that the Latin square property is not violated. We continue this process until we obtain the set S_F of filtered elements that is of a particular pre-defined size. Generally, we specify the size of S_F as a multiple of the size n of the quasigroups Q. In a second filter step we then check for the entire set S_F that its elements do not violate the desired quasigroup property P — which can in general be a combination of properties — using an automated theorem prover.

4 Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) for a quasigroup is a triple (V,D,C), consisting of (i) a finite set V of variables which represent the operator or Cayley table of the quasigroup, (ii) a finite set D, called the domain, of values that are the elements of the quasigroup, (iii) set of constraints C that assign values to the variables V from the domain D, posing constraints to represent quasigroup axioms. Quasigroups of order n can be found by considering a table of n^2 variables of the form $x_{i,j}$ where $i, j \in \{0, 1, ...n-1\}$ with possible values in the domain D of the variables where $D = \{0, 1, ..., n-1\}$ and constraining the variables of each row and column to be all different by using the constraints such as $x_{i,j} \neq x_{k,j}$ and $x_{i,j} \neq x_{i,k}$ where $i, j, k \in \{0, 1, ...n-1\}$.

The quasigroup problem is encoded in the constraint solver Minion [2], using the primal model [1]: This model has a variable $x_{i,j}$ for each cell of the Latin Square, with i and j as its coordinates (row and column). Their possible values are the elements of the quasigroup. To enforce Latin Square property i.e. the variables in the same row or column to take different values, we use the constraint in Minion [2] which uses generalized arc consistency [5] to force the variables to take different values. The following are constraints for some particular quasigroup properties:

Anti-Commutative: $x_{i,j} \neq x_{j,i}$ if and only if $i \neq j$ for all pairs $x_{i,j}$ and $x_{j,i}$.

Commutative: $x_{i,j} = x_{j,i}$ if and only if $i \neq j$ for all pairs of $x_{i,j}$ and $x_{j,i}$.

Unipotent: $x_{i,i} = x_{j,j}$ on all $x_{i,i}$ and $x_{j,j}$ where $i, j \in \{0, 1, ...n-1\}$ and n is the order of the quasigroup.

Idempotent: $x_{i,i} = i$ on all $x_{i,i}$ where $i \in \{0, 1, ...n-1\}$ and n is the order of the quasigroup.

5 Conclusion

Our experimental results demonstrate that our advanced approaches increase the solvability horizon of Minion. The following table shows the largest quasigroup with a particular property under consideration we were able to find with Minion alone and with one of our approaches:

Property	Minion	Our approach
unipotent	142	147
anti-commutative	140	143
idempotent	142	145
anti-idempotent	23	28
anti-commutative, unipotent	138	143

We are currently extending our experiments to other interesting and more complex properties of quasigroups. One possible application, is the solution of some open existence problems, for example, it is unknown if Stein quasigroups (that is, quasigroups satisfying the identity $x(xy) = yx$) exist for orders $n = 22, 23, 26, 27, 30, 34, 38, 42, 43, 46, 50, 54, 62, 66, 74, 78, 90, 98, 102, 114, 126$.

References

[1] I. Dotú, A. Del Val, and M. Cebrián. Channeling constraints and value ordering in the quasigroup completion problem. In *In Proceedings of IJCAI-03*, pages 1372–1373. Morgan Kaufmann Publishers, 2003.

[2] I. P. Gent, C. Jefferson, and I. Miguel. Minion: A fast, scalable, constraint solver. In *Proceeding of ECAI-06*, pages 98–102, 2006. IOS Press.

[3] W. McCune. Prover9. Available at `http://www.cs.unm.edu/~mccune/prover9/`.

[4] W. McCune. *Mace4 Reference Manual and Guide*. Argonne National Laboratory, 2003. ANL/MCS-TM-264.

[5] P. Shaw, K. Stergiou, and T. Walsh. Arc consistency and quasigroup completion. In *In Proceedings of the ECAI-98 workshop on non-binary constraints*, 1998.

[6] 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.

Solving Kuratowski Problems by Term Rewriting

O. Al-Hassani[1] Q. Mahesar[1] C. Sacerdoti Coen[2] V. Sorge[1]

[1] School of Computer Science, University of Birmingham
Edgbaston, United Kingdom
O.Al-Hassani@cs.bham.ac.uk, Q.Mahesar@cs.bham.ac.uk, V.Sorge@cs.bham.ac.uk
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna
Mura Anteo Zamboni, 7 (BO) Italy
sacerdot@cs.unibo.it

Abstract: We present a term rewriting system to solve a class of open problems that are generalisations of Kuratowski's closure-complement theorem. The problems are concerned with finding the number of distinct sets that can be obtained by applying combinations of axiomatically defined set operators. While the original problem considers only closure and complement of a topological space as operators, it can be generalised by adding operators and varying axiomatisation. We model these axioms as rewrite rules and construct a rewriting system that allows us to close some so far open variants of Kuratowski's problem by analysing several million inference steps.

1 Introduction

In 1922 Kuratowski asked and solved the following question on an arbitrary topological space: how many different combinations of the operators of complement and closure exist? The number turns out to be just 14 and the proof is quite small. The problem has been generalised in many different ways to consider other operators, such as union or intersection, or slightly different settings, such as point free topology (locale theory). The solution to a generalised version can be a significantly larger number of combinations, but it could also be a proof that infinitely many combinations exist. Computing finite large solutions, or obtaining an intuition for infinite variants is infeasible by hand and therefore computer automation is crucial to our solutions of the problems. Solutions or partial solutions are represented as directed graphs whose vertices are equivalence classes of provably equal combinations of operators and whose arcs represent the order relation. We present a generalised Kuratowski problem — originally proposed by Sambin in [4] — which is particularly demanding in terms of size of the approximating graphs. In order to exhibit sufficient evidence of regularities in the graph we need to compute several million edges, i.e., we need to prove several million lemmas on relations between pairs of operator combinations. We have developed a term rewriting system that models all inference rules of the problem in a uniform way and, coupled with a particular strategy and some standard graph algorithms, can show the large numbers of necessary lemmas in a few minutes. The implementation of our ideas have enabled us to prove the infinite nature of the generalised Kuratowski problem, which was up to now unknown, and serves as a basis to tackle other variants of Kuratowski's problem.

2 The Problem

Kuratowski's classical closure-complement problem [3] can be solved by observing that the following identities hold for the interior operator i, the closure operator c, and the complement '$-$' for subsets x of an arbitrary topological space. (i) $c(c(x)) = c(x)$, (ii) $--x = x$, (iii) $i(x) = -c(-x)$, and (iv) $c(i(c(i(x)))) = c(i(x))$. Applying the previous equalities only, one can show that there exist at most 14 distinct subsets one can obtain by different combinations of the three operators.

The generalisation of the Kuratowski's problem is obtained by introducing a partial order relation $\leq$ — that captures the inclusion relation for subsets — and relaxing the axioms for the operators which define the problem in a rule format, as shown below:

`reflexive:`	$x \leq x$
`anti-monotone:`	$x \leq y \rightarrow -y \leq -x$
`transitive:`	$x \leq y \wedge y \leq z \rightarrow x \leq z$
`saturates:`	$x \leq --x$
`anti-symmetric:`	$x \leq y \wedge y \leq x \rightarrow x = y$
`quasi-idempotent:`	$---x = -x$
`reduces:`	$i(x) \leq x$
`saturates:`	$x \leq c(x)$
`monotone-i:`	$x \leq y \rightarrow i(x) \leq i(y)$
`monotone-c:`	$x \leq y \rightarrow c(x) \leq c(y)$
`idempotent-c:`	$c(x) = c(c(x))$
`idempotent-i:`	$i(x) = i(i(x))$
`compatible-1:`	$c(-x) \leq -i(x)$
`compatible-2:`	$i(-x) \leq -c(x)$

Since we are effectively interested in the number of different combinations of operators that can lead to distinct sets when applied to any subset of a topological space, we define the generalised Kuratowski problem in terms of equivalent operator combinations.

Definition 1 [Generalised Kuratowski closure-complement problem] *Let $(P, \leq)$ be any partially ordered set and let $\{i, c, -\}$ be the set of operators on P axiomatised. Let $S = \{i, c, -\}^*$ be the set of all words over the operators (i.e., all possible finite combinations). We define the order relation $\leq$ over S as $w_1 \leq w_2$ iff*

$w_1(x) \leq w_2(x)$ for all $x \in P$ and $w_1, w_2 \in S$. Finally, let $\equiv$ over S be the reflexive closure of $\leq$. The generalised Kuratowski closure-complement problem then consists in computing the cardinality of $S_{/\equiv}$, the set of equivalence classes of S modulo $\equiv$.

Since the cardinality of $S_{/\equiv}$ is not necessarily finite, for practical purposes it is necessary to define finite approximations to the solution.

Definition 2 [n^{th} approximation] *Let $S_n = \{i, c, -\}^{\leq n} \subset S$ be the set of all operator combinations up to order n. For $w_1, w_2 \in S_n$ we define $\leq_n$ as $w_1 \leq_n w_2$ iff for all x $w_1(x) \leq w_2(x)$ can be shown using combinations $w \in S_n$ only (i.e., only combinations of maxiamlly n operators). Finally, let $\equiv_n$ be the symmetric closure of $\leq_n$. Then the n^{th} approximation of the generalised Kuratowski closure-complement problem is defined as computing the cardinality of $S_{/\equiv_n}$.*

The n^{th} approximation of the problem can be visually represented as a directed graph whose vertices are the equivalence classes of $S_{n/\equiv_n}$ and whose edges represent one step of the $\leq_n$ relation.

Definition 3 [Approximating graph of order n] *Let $G = (V, A)$ be a directed graph, where we define the set of vertices $V = S_n$ and the set of arcs A by $(v_1, v_2) \in A$ iff $v_1 \leq_n v_2$ for $v_1, v_2 \in V$. Now let V' be the set of all strongly connected components in G. We then define the* approximating graph of order *n as $G' = (V', A')$ where $(v_1', v_2') \in A'$ iff $v_1' \leq_n v_2'$ for $v_1', v_2' \in V'$.*

The goal is effectively to construct the graph by partitioning S_n into equivalence classes, which amounts to an inference procedure that determines if $[w_1]_{/\equiv} \leq_n [w_2]_{/\equiv}$ for $[w_1]_{/\equiv_n}, [w_2]_{/\equiv_n} \in S_n/\equiv_n$.

Theorem 4 *If the solution of the generalised problem is finite, then there exists an n such that every $(n+m)^{th}$ approximation is isomorphic (as a directed acyclic graph) to the solution.*

The theorem says that approximations *stabilise*, in the sense that larger approximations only augment the cardinality of the equivalence classes, but they do not collapse any existent distinct classes, nor do they add new arcs to the approximating graph.

The theorem does not provide an effective way to decide if an approximation is (isomorphic to) the solution.

Conjecture 5 *There exists an m such that, if for a given n the n^{th} and the $(n+m)^{th}$ approximations are isomorphic, then they are isomorphic to the solution.*

Theorem 6 *If the solution of the generalised problem is infinite, then there exists an infinite increasing sequence of approximations with larger and larger cardinalities.*

Our experience shows that in this case a clear pattern emerges, which after some time allows us to predict what new classes will be generated passing from any nth approximation to the (n+1)th approximation. This prediction can then be manually turned into a proof that these new classes will never be collapsed in later approximations and therefore the solution is infinite.

3 Implementation and Results

We have developed a bespoke Term Rewriting System (TRS) implementing the axioms of the generalised Kuratowski problem given previously [1]. The TRS has been written in pure OCaml using a graph data structure at its core. The connected-components algorithm exploits the `ocamlgraph` library [2] instantiated with an ad-hoc, optimised hashing function for equivalence classes of combinations. The TRS is fully parametric not only on the list of reduction rules, but also with respect to the words of S (i.e., the combination of operators) it generates. For the transitive reduction of the obtained graph we employ the `tred` tool and for its visualisation we use the `dot` tool.

The parametric features were particularly important for gaining intuition for the generalised Kuratowski problem, as the resulting graph is quite chaotic, in that, we were not able to find any simple description of either the set of equivalence classes or the elements of most equivalence classes. However, narrowing the elements generated in S allowed us by manual inspection of the generated graph to spot sufficient regularity to solve the problem by showing that the number of equivalence classes is infinite. In fact, all the equivalence classes whose representatives are generated by the following regular expression are distinct: $c?(--c)^*(--)?$. Moreover, each one is less than or equal to every other class generated by a longer representative (e.g. $--c \leq --c--$) and they are all bounded by $-i-$, which is also distinct from them and is the minimum of the lattice.

Although the rewriting system had been developed as a bespoke approach to solve the generalised Kuratowski problem, with its parametric implementation our procedure can be applied to a variety of related problems lying between the classical and general problem. These problems are generated by introducing the following axioms which restrict the general problem, or generalise the classical one: (i) $-- = \epsilon$, (ii) $c- = -i$, (iii) $i- = -c$, and (iv) $c-- = --c$. (v) $c = -i-$. For these problems we obtained a mixed picture of both finite and infinite cases. Applying our implementation to other problems in the domain we could quickly verify known results as well as obtain some new previously unknown results.

References

[1] O. Al-Hassani, Q. Mahesar, C. Sacerdoti Coen, and V. Sorge. A term rewriting system for Kuratowski's closure-complement problem. In *Proceedings of RTA-12*, 2012. accepted.

[2] S. Conchon, J.C. Filliâtre, and J. Signoles. Designing a generic graph library using ML functors. In *The Ninth Symposium on Trends in Functional Programming*, volume 8, pages 124–140. Intellect, 2008.

[3] C. Kuratowski. Sur l'operation a de l'analysis situs. *Fund. Math.*, 3:182–199, 1922.

[4] G. Sambin. *The Basic Picture: a structural basis for constructive topology*. Oxford University Press, 2012.

Uncertainty Modelling in Automated Concept Formation

Flaminia Cavallo Simon Colton Alison Pease

Computational Creativity Group, Department of Computing, Imperial College, London,
F.Cavallo11@imperial.ac.uk Sgc@doc.ic.ac.uk A.Pease@ed.ac.uk

Abstract: Categorisation and classification are areas that have been well studied in machine learning. However, the use of cognitive theories in psychology as a basis to implement a category formation system designed for creative purposes and based on human behaviour is still largely unexplored. Our aim in this project is to verify how some of the ideas on uncertainty and ambiguity in classification could influence concept classification in an automated theory formation system.

1 Introduction

Our research aim in this project is to investigate how influential psychological theories of human concept formation, such as those described in [8], can be interpreted for the automation of creative acts. In particular, we choose to focus on the automated theory formation system HR developed by Colton et al. [1]. This program works by combining given concepts according to a set of productions rules and heuristic measures to build a theory. Here, concepts are represented by logic predicates (definitions), and by a set of constants (examples) that satisfy these predicates. This representation corresponds to the one proposed by the classical view in the field of conceptualization in cognitive psychology [10]. As many cognitive psychologists point out, this representation is inadequate if we want to represent concepts in a human like manner because:

- There is no way to distinguish between categories' members, and to take into account the typicality of an item with respect to the category.
- It does not take into account in-between categories cases: items that partly belong to more than one category.
- It does not consider how knowledge and high-level perceptions, such as beliefs, goals and context, influence categorization.

Our aim in this project is to extend HR to enable it to operate over real word examples in a human-like way. The final scope is bidirectional: we aim to determine both how ideas suggested by the cognitive psychology community can be used to improve and extend automated concept formation techniques, and also to clarify the notions put forward in the psychology literature research by providing results and analysis from experiments undertaken.

2 HR

HR is an Automated Theory Formation program which takes a theory, conceived as a set of initial concepts, and applies a set of production rules on it in order to construct new concepts. These production rules take as an input the definition of one or two concepts and output the definition for the new concept. For example, the match production rule equates two variables in a definition, and the negate rule negates certain clauses in a definition. Once the new concept definition is created, HR calculates the success set of the definition by collating all tuples of objects which satisfy the definition. The set of positive examples is then used to make conjectures about the new concept, in the form of equivalence conjectures, implication conjectures, or non-existence conjectures. Conjectures are either proved by the OTTER theorem prover [5] or rejected because of a counterexample found by the MACE model generator [6]. The theory is then enriched with either the new theorem or with the newly found counter-examples. HR follows a best-first non-goal-oriented search. This is dictated by an ordered agenda and a set of heuristic rules used to evaluate the interestingness of each concept. The scope of HR is to form interesting clausal theories, starting with some minimal knowledge and enriching it by performing both inductive and deductive reasoning.

3 Project Outline

According to the prototype view in the field of cognitive psychology, humans categorize an item by comparing it with each known category's most typical item (real or imaginary), which is also called a prototype [3]. The similarity between each pair is used to determine the new object's typicality with respect to every category and can be interpreted as a measure of how much the item belongs to the respective categories. Prototypes are represented as schemata reporting the features which we gradually learn are the most frequent and relevant to each category. Hence prototypes are flexible entities and they are influenced by the categorization process itself. We have decided to explore how these observations can influence category creation and theory formation by including a similar notion into HR. To do so, we will assign a degree of membership to every tuple of objects constituting an example of a concept. This parameter will represent the percentage with respect to which the example belongs to the concept, and hence will be directly proportional to its typicality. Note that this implies that an item does not need to fully belong to just one category - in-between categories cases are allowed. The idea is similar to the one used in fuzzy logic, where the degree of membership of an item is determined by a membership function. However, in our case, this membership function will be flexible over time, depending on the already classified category members and modified every time a new item is classified within a category.

4 Calculation of Typicality

Dunmore [2] observed how concept definitions are chosen and developed according to their use by presenting a study on the definition of prime numbers. This concept was initially defined as "a number which is only divisible by 1 and itself". The number 1 satisfies this definition, and hence it is considered a positive example. However, 1 constitutes a counterexample to many conjectures about primes, for example the Fundamental Theorem of Arithmetic, stating that every natural number is either a prime or can be represented uniquely as a product of primes. In the current version of HR, the above counterexample would be enough to make this conjecture false. A different approach is to review the concept definition of prime numbers itself, for example by considering a different definition such as "number with exactly two divisors". Another example has been proposed by Lakatos [4] who, in the attempt to prove Euler's conjecture, reported five different definitions for polyhedra.

In our system, we will allow a concept to have multiple definitions. In the examples above, prime numbers would have two definitions and polyhedra would have five definitions. The typicality of an item with respect to a concept will then be calculated according to three measures:

- The number of concepts' definitions that the item satisfies. In the prime number case, 1 satisfies 50% of the definitions, 3 satisfies all definitions and 4 satisfies no definition. In the polyhedra case, the only polyhedra that satisfy all five definition are the regular polyhedra.

- The amount of tweaking that each definition would need in order to include the item. In the prime number example above, the definition "number with exactly two divisors" could be modified to "number with exactly one divisor" or "number with maximum two divisors" in order to include 1. The amount of tweaking will be measured by the number of HR's production rules that would be involved in modifying the definition. The salience of each part of the definition will also be taken into consideration. To calculate it we will take inspiration from psychological studies that underline the importance of both the relevance and the familiarity of attributes in a similarity task [9, 7].

- The number of conjectures about the concept that the item supports, in a similar way to what Lakatos suggested in [4]. In the example above, the typicality of 1 will decrease as we discover that 1 is a counterexample of the Fundamental Theorem of Arithmetic.

The typicality measures over a category success set would help us recognize that conjectures like the Fundamental Theorem of Arithmetic are probably true, as they are true for most typical examples. Moreover, these observations will change our beliefs on what the correct definition of a prime number is. The uncertainty over these beliefs would then be used as new properties about the concept are discovered, and as new more complicated concepts are constructed.

5 Conclusion

This project is still at a preliminary stage. However, we can see it leading to results that can be applied in different areas. For example, the program could be used as a learning and data-mining system on large datasets representing human behaviours which are classifiable, whose properties are not known, and to which a computer could actively contribute in a creative way. A possible application follows the lines of the example given above: the study of how mathematical concepts, conjectures and proofs gradually evolve as more things are discovered about them.

References

[1] S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag New York, 2002.

[2] C. Dunmore. *Meta-level revolutions in mathematics*. In Gillies, D., editor, Revolutions in Mathematics, pages 209225. Clarendon Press, Oxford., 1992.

[3] Rosch E. Natural categories. *Cognitive Psychology*, 4(3):328 – 350, 1973.

[4] I. Lakatos. *Proofs and Refutations*. CUP, Cambridge, UK, 1976.

[5] W. McCune. *The OTTER user's guide*. Technical Report ANL/90/9, Argonne National Laboratories, 1990.

[6] W. McCune. *A Davis-Putnam program and its application to finite first-order model search*. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.

[7] D. L. Medin and J. G. Bettger. Presentation order and recognition of categorically related examples. *Psychonomic Bulletin and Review*, 1:250–254, 1994.

[8] G. Murphy. *The Big Book of Concepts*. MIT Press, 2002.

[9] A. Ortony. Beyond literal similarity. *Psychological Review*, 86(3):161–180, 1979.

[10] E. E. Smith and D. L Medin. *Categories and concepts*. Cambridge, MA: Harvard University Press, 1981.

METTEL^2: Towards a Prover Generation Platform

D. Tishkovsky R. A. Schmidt M. Khodadadi

The University of Manchester, {dmitry,schmidt,khodadadi}@cs.man.ac.uk

Abstract: This paper introduces METTEL^2, a tableau prover generator producing JAVA code from the specifications of a logical syntax and a tableau calculus. It is intended to provide an easy to use system for non-technical users and allow technical users to extend the implementation of generated provers.

1 Introduction

The tableau method is one of the most popular deduction approaches in automated reasoning. Tableau methods in various forms exist for various logics, and many tableau provers have been implemented. Based on this collective experience in the area, our recent research has been concerned with trying to develop a framework for synthesising tableau calculi from the specification of a logic or logical theory. The tableau synthesis framework introduced in [5] effectively describes a class of logics for which tableau calculus synthesis can be done automatically. This class includes many modal, description, intuitionistic and hybrid logics. Our long-term goal is to synthesise not only tableau calculi but also implemented tableau provers.

As a step towards this goal, we have implemented a tool, called METTEL^2, for automatically generating an implemented tableau prover from the specification of a set of tableau rules provided by the user. METTEL^2 is the successor of the METTEL tableau prover [7, 1]. METTEL and other existing generic tableau provers such as LOTREC [4] and the Tableaux Work Bench (TWB) [2] do not produce code for a prover but rather act as virtual machines that perform tableau derivations.

METTEL^2 considerably extends METTEL functionalities. METTEL^2 generates JAVA code for a tableau prover to parse problems in the user-defined syntax and solve satisfiability problems. In order to come closer to the vision of a powerful prover generation tool, METTEL^2 is equipped with a flexible specification language. This allows users to define their logic or logical theory with their syntactic constructs. Thus no logical operators are predefined in METTEL^2. The generated tableau provers can be tuned further. Addressing the needs of an advanced user, an API of the tableau core engine is designed to accept user-defined tableau expansion strategies implemented as JAVA classes.

Compared with the previous METTEL system, the tableau reasoning core of METTEL^2 has been completely reimplemented and several new features have been added, the most important being: dynamic backtracking and conflict-directed backjumping, ordered forward and backward rewriting for operators declared to be equality and equivalence operators. There is support for different search strategies and rule application strategies. To our knowledge, METTEL^2 is the first system with full support of these techniques for an *arbitrary* logical syntax.

2 Language and tableau calculus specification

The language in METTEL^2 for specifying the syntax of a logical theory is in line with the many-sorted object specification language of the tableau synthesis framework defined in [5]. Following is a simple 'non-logical' example for describing and comparing lists.

```
specification lists;
syntax lists{
   sort formula, element, list;
   list empty = '<>' |
          composite = '<' element list '>';
   formula elementInequality =
     '[' element '!=' element ']' |
      listInequality = '{' list '!=' list '}';
}
```

The first line starting with the keyword `specification` defines `lists` to be the name of the user-defined logical language. The `syntax lists{...}` block consists of a declaration of the sorts and definitions of logical operators in a simplified BNF notation. Here, the specification is declared to have three sorts. For the sort `element` no operators are defined. This means that all `element` expressions are atomic. The second line defines two operators for the sort `list`: a nullary operator `<>` (to be used for the empty list) and a binary, infix operator `<..>` (used to inductively define non-empty lists). `composite` is the name of the operator `<..>`, which could have been omitted. The rest defines two types of inequality as expressions of sort `formula`. The first mentioned sort in a declaration, in our case `formula`, is the *main sort* of the defined language.

The tableau rule specification language of METTEL^2 is loosely based on the tableau rule specification language of METTEL, extended with rule *priority value*. Smaller priority values imply a rule has higher priority.

Tableau rules for list comparison might be defined as follows.

```
[a != a] / priority 0$;
{L != L} / priority 0$;
{<a L0> != <b L1>} / [a != b] $|
                    {L0 != L1} priority 2$;
```

As the parsing of rule specifications is context-sensitive the various identifiers (`a`, `L`, `L0`, etc) are recognised as symbols of the appropriate sorts. Thus *sorts* of identifiers are distinguished by their context and not their case. The first two rules are closure rules since the right hand sides of the `/` are empty. They reflect that inequality is irreflexive. The last rule is a branching rule.

3 Prover generation

The parser for the specification of the user-defined logical language is implemented using the ANTLR parser generator. The specification is parsed and internally represented as an abstract syntax tree (AST). The internal ANTLR format for the AST is avoided for performance purposes. The created AST is passed to the generator class which processes the AST and produces the following files: (i) a hierarchy of JAVA classes representing the user-defined logical language, (ii) an object factory class managing the creation of the language classes, (iii) classes representing substitution and replacement, (iv) an ANTLR grammar file for generating a parser of the user-specified language and the tableau language, (v) a main class for the prover parsing command line options and initiating the tableau derivation process, and (vi) JUNIT test classes for testing the parsers and testing the correctness of tableau derivations.

The generated JAVA classes for syntax representation and algorithm for rule application follow same paradigm as in the old METTEL system [7].

METTEL^2 implements two general techniques for reducing the search space in tableau derivations: dynamic backtracking and conflict directed backjumping. Dynamic backtracking avoids repeating the same rule applications in parallel branches by keeping track of rule applications common to the branches. Conflict-directed backjumping derives conflict sets of expressions from a derivation. This causes branches with the same conflict sets to be discarded.

The core tableau engine METTEL^2 provides various ways for controlling derivations. The default search strategy is depth-first left-to-right search. Other strategies can also be implemented and passed to the core.

The rule selection strategy can be controlled by specifying priority values for the rules in the tableau specification. Rules with the same priority values are iterated sequentially. To ensure fairness all applicable rules within the same priority group are queried for applications an equal number of times. Preference is given to rules from groups with smaller priority values. Again the user could implement their own rule selection strategy and modify the generated code.

Blocking in tableau derivations can be implemented as variants of the unrestricted blocking rule [5]. The unrestricted blocking rule ensures termination of a sound and complete tableau calculus in case the specified logic has the finite model property (cf. [5, 6]).

The binary version of METTEL^2 is available for download from [1] as a **`jar`**-file. A web-interface for METTEL^2 is also provided, where users can generate a prover by entering their specifications and tableau calculus. The user can then either download the generated prover as a **`jar`**-file or directly run the generated prover in the interface.

Several test cases have been prepared for the system covering a variety of logics including Boolean logic, modal logic S4, description logics $\mathcal{ALCO}$ and $\mathcal{ALBO}^{\mathsf{id}}$ [6], a hybrid logic with graded modalities and linear-time temporal logic, with or without capacity constraints [3]. Sample specifications with unrestricted blocking are the tableau calculi for S4, $\mathcal{ALBO}^{\mathsf{id}}$ and linear-time temporal logic (with constraints). Some of these test cases and the **`lists`** example from this paper (as well as an extended version with a concatenation operator) are available at [1].

4 Concluding remarks

METTEL^2 and METTEL are small but essential steps to the very ambitious goal to create a reliable and easy to use prover generation platform which implements the automated synthesis framework [5]. We do not aim to provide a sophisticated meta-programming languages, but to provide easy to use systems, for non-technical users, and for technical users, expandable systems by allowing them to write their own JAVA classes and integrate them using the provided API. More information about how to generate a prover using METTEL^2 is available at [8].

References

[1] METTEL website. `http://mettel-prover.org`.

[2] P. Abate and R. Goré. The tableau workbench. *Electronic Notes in Theoretical Computer Science*, 231:55–67, 2009.

[3] D. Dixon, B. Konev, R. A. Schmidt, and D. Tishkovsky. A labelled tableau approach for temporal logic with constraints. Manuscript, submitted for publication, available at `http://mettel-prover.org/papers/dkst12.pdf`, 2012.

[4] O. Gasquet, A. Herzig, D. Longin, and M. Sahade. LoTREC: Logical tableaux research engineering companion. volume 3702, pages 318–322, 2005.

[5] R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. 7(2:6):1–32, 2011.

[6] R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. Manuscript, available at `http://mettel-prover.org/papers/ALBOid.pdf`, 2011.

[7] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. METTEL: A tableau prover with logic-independent inference engine. volume 6793, pages 242–247, 2011.

[8] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. Mettel2: Towards a prover generation platform. Manuscript, submitted for publication, available at `http://mettel-prover.org/papers/MetTeL2SysDesc.pdf`, 2012.

Index of Authors