Ninth Workshop on Automated Reasoning

Bridging the Gap between Theory and Practice

Collected Abstracts

Toby Walsh (editor)
Cork Constraint Computation Center
University College Cork
Ireland

OAV-VVT Expert
Integrated Verification and Validation Tool
For Knowledge Base System
"Based on OAV Representation Knowledge Technique"

ABDOLLAHZADEH-BARFOROUSH Ahmad DADKHAH Chitra

Amirkabir University of Technology
Computer Engineering Department
Tehran –Iran, Hafez Avenue
E-mail: (Ahmad, Dadkhah) @ce.aku.ac.ir

Abstract:
Verification and Validation (V&V) are the most important issues in building knowledge base systems. Researchers and practitioners in the field of AI applications generally all agreed that Knowledge Base in any intelligent systems must be adequatatly verified and validated. In this paper we present OAV-VVT as an verification and validation expert system for V&V during the implementation and maintenance phases of Knowledge Base system life cycle.
The approach, which has been chosen in OAV-VVT system, is based on knowledge representation and reasoning techniques. Object-Attribute-Value (OAV) Knowledge representation method has been used in OAV-VVT expert. All common issues of V&V such as consistency, completeness, semantic and logical contradiction and correctness of knowledge base have been considered in OAV-VVT expert. The designed system is application independent: it has several important advantages:
First, OAV-VVT itself is an expert system with reasoning and explanation mechanisms.
Second, OAV-VVT can be either used during the knowledge representation phase or refinement of knowledge base.
Third, OAV-VVT is application and knowledge base representation independent: so it can be used verifying and validating all AI systems based on different knowledge base representation techmics (i.e.: logic, semantic network, frame, production rule, predicate logic).
Keyword: Verification and Validation, Knowledge base, Knowledge representation, OAV triple, Expert systems, AI systems.

1:Introduction

the lack of adequate and precise knowledge is a limiting factor to build an efficient AI application. When intelligent methods are used in real applications, it is an important factor to be able to check the consistency of knowledge and the correctness of the system's reasoning. As with any software system or application, attention to quality and safety must be paid throughout the development of knowledge base systems. The need for an integrated approach towards Verification and Validation (V&V) of Knowledge Base (KB) is quite obvious. In this paper we present OAV-VVT as an verification and validation expert system for V&V of KB. In order to built such an expert system tool, first of all, it is necessary to have a common definition of V&V. The table. 1 is indicateing an over-view of several researches and practitioners defined V&V of KB in several ways at different times.

Table. 1: The definitions of V&V by the researchers and practitioners.

Researcher's Name	Validation	Verification
BOEHM [Boe84]	Building the right system.	Building the system right.
IGNIZIO [Ign91]	. Justification for the employment of an expert system. . The verification of the overall performance of the expert system.	The validation of the consistency and completeness of the expert system's rule base.
MENSHOEL & DELAB [Del&Men93]	Addresses software system's usefulness with respect to some real-world task, regardless of its specification.	Concerns a software system's conformance to its specification.
KANDEL & SMITH [Smi&Kan93]	A test of whether the ES matches the design ideas, i.e., whether it matches the technical requirements and expectations.	Examine the technical aspects of an ES in order to determine whether the ES is built correctly.
WU & al [Wu&al94]	The correctness of inference without considering the Input/Output.	The correctness of systems outputs according to specific Inputs.
ROUSSET & al [Rou&al96]	The correctness of system's output depends on Test cases and Initial valid Fact Base.	Checking the logical and semantic contradiction of Rule Base.
CARDENOSA & ESCORIAL [Car&Esc99]	The set of activities in charge of checking that the system is adapted to the environment and user requirements.	The set of activities aiming at checking that the system is adapted to the system requirements.
KNAUF & al [Kna&al00]	Asks whether or not a system is considered to be the required one, something that somehow lies in the eyes of the beholder.	Provides a firm basis for the question of whether or not a system meets its specifications.

The approach, which has been chosen in OAV-VVT system, is based on knowledge representation and reasoning techniques. OAV-VVT system is able to V&V the knowledge of KB with the Object-Attribute-Value (OAV) knowledge representation technique. The existing KB will transfer to OAV knowledge representation technique (canonical KB). We will show that how the major techniques of knowledge representation (i.e. logic, semantic network, frames, production rule) could be transfered to OAV. OAV-VVT expert's reasoning validates and verifies the canonical KB. The dynamic process of V&V in OAV-VVT expert will modify all knowledge in case of any error and construct a new knowledge base. The result will recognize a valid KB. Figure .1 shows the process of incoming knowledge to OAV-VVT system.

A solution to the problem of contradiction in knowledge discovery applications.

David Anderson
School of Computer Science & Mathematics
University of Portsmouth
Portsmouth PO1 2EG

cdpa@btinternet.com

Abstract

There have been a number of very credible attempts to devise paraconsistent logics to deal with the problems caused by the unavoidability of contradictions in knowledge bases and elsewhere. This paper suggests a set of principles for generating logical operator semantics which are broadly drawn from classical logic. These lead directly to a paraconsistent logic, LM_4, which significantly outperforms other systems and solves the problem for all practical purposes without giving rise to the difficulties inherent in other attempted solutions

Towards automated generation of beliefs in BDI agents

A. Basukoski, A. Bolotov
Harrow School of Computer Science, University of Westminster,
Harrow, Watford Road, HA1 3TP.
E-mail: {A. Basukoski,A.Bolotov}@westminster.ac.uk

Abstract

Formal BDI frameworks such as LORA [5] which use temporal logic as a tool for describing beliefs about the environment consider an agent behavior in the following control loop:

```
Algorithm: Agent Control Loop
1. while true
2.     observe the world;
3.     update internal world model;
4.     deliberate about what
          intention to achieve next;
5.     use means ends reasoning to
           get a plan for the
            next intention;
6.     execute the plan;
7. end while.
```

Whereas the vast majority of agent literature focuses on steps 4, 5, and 6, we are interested in the transition 2→3 which forms a link between the perceived world and the way knowledge about that world is represented within the BDI framework. One of the efficient ways of specifying beliefs is to use combinations of temporal or dynamic logics and modal logics (see, for example, [1, 3]). We define a formal structure from which Beliefs can be automatically extracted in a way suitable for such formal representation.

Thus, we introduce an algorithm to incrementally generate this structure, which we call a 'A Temporal Lattice'. We invoke the standard technique of the Formal Concept Analysis [4, 2], where an observation, ρ, is formally represented as $\rho \subseteq E \times I$, where E is a set of *extents*, and I is a set of *intents* [2]. Since ρ can be viewed as an expression in propositional logic [2], the set of observations forms the alphabet, $\mathcal{A}$ for labelling the nodes in a graph being constructed.

At each step, we assume that the deliberation on which action (from the set of possible actions) must be next taken is done by a deliberation function which forms part of the BDI architecture. Thus, viewing this deliberation function as a transition function, we unwind the structure as a graph such that, given a state, n, we make a non-deterministic choice of the successor state, m.

This forms a sequences of states $\tau_0, \tau_1, \tau_2, \ldots$, which are linked by actions $act \in Act$. Additionally, extending the alphabet $\mathcal{A}$ by **true** and **false** (with their standard meaning accepted in classical logic) we label them by the corresponding expressions from $\mathcal{A}$ (the initial state is labelled by **false**). We refer to this sequence as level 0 ($n = 0$) of the Temporal Lattice to be constructed at the next stage.

Given a sequence τ, (see Figure 1) the initial node of the Temporal Lattice, is the initial state of τ, and we denote this node as $\omega_{0,0}$, indicating that it occurs at the 0-th point of time and 0-th level. The successor node, $\omega_{1,0}$ is the state τ_1 of τ and has the same label as the state τ_1, say, ρ_k. Now, we build the node $\omega_{1,1}$ such that its label, ρ_l, satisfies the following condition:

$$\rho_l \equiv (\mathbf{false} \vee \rho_k) \equiv \rho_k$$

and the node $\omega_{1,-1}$ such that its label, ρ_m, satisfies the following condition:

$$\rho_m \equiv (\mathbf{false} \wedge \rho_k) \equiv \mathbf{false}.$$

The second index of the nodes $\omega_{1,1}$ and $\omega_{1,-1}$ indicate their level in the lattice, i.e. level 1 and level -1,

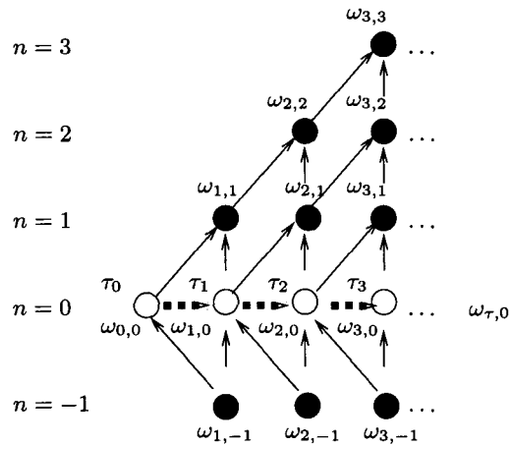

Figure 1: Temporal Lattice

respectively. Repeating this procedure again on subsequent nodes, we derive the Temporal Lattice (the fact the structure is indeed a lattice follows from the construction algorithm).

The analysis of the Lattice enables us to

- generate abstractions,
- extract expressions (in linear-time temporal logic) for the deliberation function,
- derive models of linear computations where these formulae are satisfied.

We believe that the Temporal Lattice is an efficient method (at every time point, i, we generate at most $2 \times i$ nodes in the lattice) to generate Knowledge and Beliefs incrementally from raw percepts and is therefore useful in agent applications where

1. there is limited perception of the environment
2. there is no prior knowledge about the environment
3. and the environment is dynamic and nondeterministic.

References

[1] C. Dixon, M. Fisher, and A. Bolotov. Clausal Resolution in a Logic of Rational Agency. *Artificial Intelligence*, 2002. To appear.

[2] B. Gantner and R. Wille. *Formal Concept Analysis.* Springer, 1996.

[3] U. Hustadt, C. Dixon, R. Schmidt, M. Fisher, J. Meyer, and W. van der Hoek. Reasoning about agents in the karo framework. In *Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning (TIME-01)*, pages 206–213, Cividale del Friuli, Italy, June 2001. Computer Society Press.

[4] J.F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Brooks Cole Publishing Co, 1999.

[5] M. Wooldridge. *Reasoning about Rational Agents.* MIT Press, July 2000.

Agent-Based Theorem Proving

Christoph Benzmüller
Fachbereich Informatik
Universität des Saarlandes
chris@ags.uni-sb.de

Volker Sorge
School of Computer Science
University of Birmingham
V.Sorge@cs.bham.ac.uk

1 Introduction

We report on a commencing project that is concerned with the transfer of techniques from multi-agent systems research to the domain of automated reasoning. The goal is to achieve more flexible organisation within a theorem-prover, not only to provide a better approximation to the human reasoning approach, but also to solve more complex theorems.

In previous work we have already experimented to some extend with the transfer of multi-agent techniques to theorem proving. We have developed a distributed architecture to support interactive theorem proving [3] within the ΩMEGA theorem-proving environment [1]. Its automation has led to a prototype distributed reasoning system [2], which comprises several hundred independent reasoning processes of which some encapsulate full reasoning agents such as independent automated theorem provers, model generators, and computer algebra systems. The cooperation of the single components is achieved via a central hierarchical blackboard architecture and the overall proof is constructed within a centralised proof object. A significant bottleneck is the problem of communicating partial proofs between the different systems involved.

Many of our problems are typical for distributed problem solving environments and have been tackled with some success in field of multi-agent systems [6; 4]. Here single agents are comprised of autonomous computational entities that have both self perception and a perception of their environment containing other agents. This enables them to independently pursue their goals as well as to flexibly form societies with other agents in order to cooperatively achieve goals. Some of the techniques we try to incorporate from by multi-agent systems research are, for instance, ways to negotiate between agents, efficient communication languages, distribution of limited resources and heuristics to evaluate performance.

The main goal of our project is to build a flexible automated theorem proving system based on the agent paradigm that enables us to tackle hard problems by cooperation within a society of heterogeneous reasoning agents. The single agents are independent from each other but can communicate and cooperate in order to produce a proof. Thereby we will exploit the considerable insights we have gained in our previous work when we experimented with an agent-based approach in ΩMEGA. Currently, we extend our approach, bringing together work on distributed theorem-proving, the development of societies of agents, and the modelling and evolution of cooperation, in order to investigate how an *agent-oriented* approach may be used to improve theorem-proving. We will, in particular, tackle the shortcomings of the predecessor system, such as the communication bottleneck, the limited distribution, and the lack of autonomy of the reasoning agents.

In the following we present some of the major research tasks of the project.

2 Distributed proof search

The main aim of the agent-based system is to enable the distribution of the proof search among groups of reasoning agents. There are two possible ways to distribute this search: (1) to tackle one subproblem using several reasoning agents concurrently or cooperatively, and (2) to work on different subproblems in parallel. This essentially corresponds to *and-or-parallelism* within the search.

From the theorem proving point of view, this type of search poses several challenging problems. (1) The identification of possible *dependencies between parallel proof attempts*: Which agents might interfere with each other and how? What is the relevant information that needs to be exchanged while tackling different subgoals in parallel? When should the parallel proof attempts by resynchronised? (2) The problem of *backtracking*: How can backtracking be organised in parallel proof attempts with heterogeneous reasoning agents? Which are the proof states the system can backtrack to and what kind of information, such as logical dependencies between different proof branches, can be exploited? What are the criteria for backtracking? (3) The construction of a joint *uniform proof object*: If a proof attempt has been successful we are still interested in the proof that was constructed, i.e. it is necessary to reconstruct a uniform proof object from the results of the distributed search. For this the agents involved have to retain some information about their own search. Therefore, we have to investigate appropriate granularity of this information and how should it be represented, communicated, and maintained? A final uniform proof object is also particularly important in order to *guarantee correctness* of the proof. Since very heterogeneous systems can be involved (for in-

stance, provers for classical and constructive logic) assembly of a distributed proof might not lead to a correct proof.

3 Flexible cooperation

The major means to achieve a flexible behaviour of the system is to enable the *dynamic formation of clusters* of reasoning agents. These clusters can be of various types. For example, agents that complement each other can attempt to cooperatively solve a problem (for instance, the cooperation of a first order and higher order theorem prover). Agents with similar abilities can form clusters that have a certain reasoning expertise and can work concurrently on given problems (for instance, a collection of first order theorem provers). This increases the likelihood that a problem at hand will be solved by one of the agents. Clusters can also be comprised of agents with contrasting abilities (for instance, an ATP and a model generator), which would enable the classification of some problems, for example, an assertion is a theorem or has a counter model. In order to form clusters it is necessary that the agents have some knowledge of their own and each others abilities.

In our prototype system, a significant bottleneck was the problem of *communicating partial proofs* between the different systems involved — in some cases, the communication time outweighed the actual time needed for proof search. Therefore, one major concern will be to find a concise representation for both problems and proofs in order to communicate them more efficiently. Currently, all communication is routed via a central structure in a uniform format, a higher-order natural deduction calculus. However, this is not necessarily desirable as certain agents may be able to communicate more concisely between each other, for instance, by exchanging sets of clauses. Therefore, we aim at communication languages in which relevant elements of the proof process can be effectively transferred.

4 Resources

We have to *identify the resources* that are limited and that have to be managed in the agent-based system and to develop resource-guided heuristics for spawning new threads within the theorem-prover. Thereby we will consider as resources not only the classical resources such as computation time or memory, but also those special to the domain of automated reasoning. These additional resources have generally to do with logical dependencies between parallel proof attempts. For example, we can consider the instantiation of a variable in the proof as a resource. If one agent wants to instantiate this variable with a particular term it can affect the proof search of other agents. Therefore, the resource would be consumed by the agent which will need to be broadcasted to the other agents, or the agent might even have to negotiate first to obtain the right to instantiate this variable. Another sensible resource may consist in the *size* of the representation of the partial proofs exchanged between the agents.

5 Architecture for Reasoning Agents

With the background of the above requirements for our system we have to investigate what understanding our reasoning agents need to have of themselves and of other agents. The former is necessary for an agent to gain sufficient self-recognition to know its strengths and weaknesses and to judge its abilities to contribute to the solution of a given problem. The latter is generally important for the exchange of information between agents and in particular helpful in negotiating with other agents about forming societies that cooperatively pursue a task. For both cases our agents need to have heuristics to judge themselves and others. These heuristics can be influenced by a performance measure for past runs and which can vary for different application domains.

Therefore, will investigate SimAgent [5], a very flexible toolkit for exploring agent architectures. This will be studied in order to decide which of its features may be useful for the purposes of an agent-based theorem prover. Examples might include mechanisms for controlling resource allocation, mechanisms for interfacing symbolic condition-action rules running within an agent's cognitive system with "lower level" mechanisms treated as "black boxes", and mechanisms for linking agent behaviours with graphical displays. We also want to investigate to which extend already existing networks of mathematical reasoning systems can be integrated or exploited in our framework.

References

[1] Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Karsten Konrad, Erica Melis, Andreas Meier, Wolf Schaarschmidt, Jörg Siekmann, and Volker Sorge. ΩMega: Towards a Mathematical Assistant. In William McCune, editor, *Proceedings of CADE-14*, volume 1249 of *LNAI*, pages 252–255, Townsville, Australia, 1997. Springer Verlag, Germany.

[2] Christoph Benzmüller, Mateja Jamnik, Manfred Kerber, and Volker Sorge. Experiments with an Agent-Oriented Reasoning System. In Franz Baader, Gerhard Grewka, and Thomas Eiter, editors, *KI 2001: Advances in artificial intelligence : Joint German/Austrian Conference on AI*, volume 2174 of *LNAI*, pages 409–424, Vienna, Austria, 2001. Springer Verlag, Germany.

[3] Christoph Benzmüller and Volker Sorge. Ω-ANTS – An open approach at combining Interactive and Automated Theorem Proving. In Manfred Kerber and Michael Kohlhase, editors, *Symbolic Computation and Automated Reasoning - The CALCULEMUS-2000 Symposium*, pages 81–97, St. Andrews, United Kingdom, 2001. AK Peters, Natick, MA, USA.

[4] Nicholas R. Jennings and Michael J. Wooldridge, editors. *Agent Technology: Foundations, Applications, and Markets.* Springer Verlag, , Germany, 1998.

[5] A. Sloman and B.S. Logan. Building cognitively rich agents using the sim_agent toolkit. *Communications of the Association for Computing Machinery*, 43(2):71–77, 1999.

[6] Gerhard Weiss, editor. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1999.

Solving Multiple Containment Tests for Linear Constraints over Integers

Dmitri Chubarov* and Andrei Voronkov

Department of Computer Science
The University of Manchester

ABSTRACT

We consider the containment problem for sets integral solutions of systems of linear inequalities

$$\left\{ x \in \mathbb{Z}^k \,\middle|\, Ax \leq b \right\}.$$

Given two such sets Φ and Ψ, the Containment Problem as a decision problem is to tell if $\Phi \subseteq \Psi$. Many algorithms for model-checking of infinite-state systems require solving multiple instances of the problem when the same set Φ is to be tested for containment in different sets $\Psi_1 \ldots \Psi_k$. This is a different problem. We describe a way to represent a set Φ based on the computation of a Hilbert basis, such that containment in a set Ψ, represented with a system of linear inequalities, can be tested in linear time in the size of the basis and inequalities.

The representation used in the model-checker BRAIN [RV02] can be described as follows. The set Φ of integral solutions for a system of linear inequalities $Ax \leq b$ can be decomposed into a cone of solutions of the homogeneous part $C = \left\{ x \in \mathbb{Z}^k \,\middle|\, Ax \leq 0 \right\}$ and a finite set of vectors B, such that $\Phi = C + B$. The cone C has a finite Hilbert basis H. Since H and B are computed, containment test for Φ and Ψ can be performed by testing if each vector of $H \cup B$ satisfies Ψ. However the number of vectors in $H \cup B$ can be exponential in the size of the system (A, b).

The usual alternative is to apply the methods of integer programming to check that Ψ is consistent with the negation of every inequality which defines Φ. Testing consistency of a system of linear inequalities over integers is an NP-complete problem.

Depending on the number of containment tests, the performance gain on solving containment becomes worth the effort spent in computing the Hilbert basis. We observed the implementation which uses Hilbert bases to perform better when applied to model-checking for a class of infinite-state transition systems.

The model-checking problem is defined by a temporal logic and a class of transition systems. It is the problem to decide if a given temporal formula holds

* supported by an Overseas Research Students Award

over the sequences of states of the given transition system visited during the evolution of the system.

We consider a particular class of transition systems with states represented by integral vectors. Even for EF formulas which specify safety properties, the model-checking problem is undecidable. However, there exists a semi-decision procedure which decides model-checking for some special classes of systems [EFM99]. The procedure is based on the *backward reachability* algorithm.

The algorithm computes fixed points of the inverse transition relation. The inverse transition relation is the relation between states and all their possible predecessors. Instead of operating on individual states the algorithm operates simultaneously on sets of states defined by systems of linear inequalities. This allows to apply model-checking techniques to infinite state spaces and evade the "state explosion". For example, computation of a least fixpoint of a binary relation is an iterative process which generates an increasing sequence of sets of states. Each set is represented as a finite set of systems of linear inequalities. Each time a new system is generated it is checked for containment with the systems generated before, which results in multiple checks with the same system.

This pattern of behaviour is exhibited by the backward reachability algorithm, when it was applied to models of uniform cache-coherency protocols and broadcast protocols.

We have compared an algorithm used in BRAIN which computes Hilbert bases with a Gauss-Jordan solver of CLP(Q) on problems where rational and integer containment tests produce the same result and observed better results from the implementation from BRAIN. In our experiments we follow the methodology of using sequences of problems extracted from practical application [HNRV]. Preliminary results show that representing solutions of linear inequalities with Hilbert bases gives a significant performance gain on many reachability problems.

References

[RV02] Rybina, T., Voronkov, A.: Using canonical representations of solutions to speed up infinite-state model checking. Submitted

[EFM99] Esparza, J., Finkel, A., Mayr, R., On the Verification of Broadcast Protocols. Proceedings of LICS'99, pp. 352–359

[HNRV] Nieuwenhuis, R., Hillenbrand, T., Riazanov, A., Voronkov, A., On the Evaluation of Indexing Techniques for Theorem Proving, IJCAR'01, LNAI 2083, pp. 257–271

On learning typed-unification grammars

Liviu Ciortuz

CS Department, University of York
Heslington, York, YO10 5DD, UK, E-mail: ciortuz@cs.york.ac.uk

Interest in typed unification grammars in Natural Language Processing can be traced back to the seminal work on the PATR-II system [13]. Since then, the *logics* of feature constraints were studied and became well-understood [15] [3], and different types of unification-based *grammar formalisms* were developed by the Computational Linguistics community — most notably Lexical Function Grammars [6], Head-driven Phrase Structure Grammars (HPSG, [11]) and Categorial (unification) Grammars [17]. During the last years *large-scale implementations* for such grammars were devised.[1] More recently, important advances in the elaboration of techniques for efficient unification and parsing with such grammars [16]. Up to our knowledge, the possibility of automate learning/development of such grammars is still (largely) unexplored.

We elaborated two procedures to be used in automatic learning of typed-unification grammars, namely *specialisation* and respectively *generalisation* of typed feature structure (FS) definitions. Their *aim* is to make an input set of sentences get a better coverage w.r.t. the given grammar.

In specialising a grammar, new atomic constraints are added to a certain feature path value (i.e. substructure in a typed FS), in such a way that sentences (or parses) which were wrongly accepted in the form of the grammar be finally rejected (by the learnt form of the grammar). Reversely, in generalising a grammar, parses (or more generally, sentences) which are (wrongly) rejected by the input the grammar are forwarded to the learning system, which makes their (partial) analysis provide clues about which constraints in the type FS definitions have to be eliminated or softened in order to make that parse become acceptable by the (new form of the) grammar. The two — specialisation and generalisation — procedures combine naturally into a unified learning strategy.

The *approach* we followed in developing the two procedures for learning typed-unification grammars is that proposed by Inductive Logic Programming (ILP) [9]. However, instead of applying it to a first-order translation of the grammar to be learnt, we adapted the ILP schema to the feature constraint (subset of the first-order) logic specific to such grammars [1] [3] [2].

As *implementation*, our prototype system for learning typed-unification grammars was developed by extending the LIGHT compiler system [4] with an ILP-like module. Experiments were successfully conducted on a grammar adapted from [12], which embodies the main principles of the HPSG theory.

Currently, in order to *scale-up* the range of application for the learning procedures we developed, we study a restricted form of generalisation of typed FSs (we call it F-generalisation) on LinGO, large scale HPSG grammar for English developed at CSLI, University of Stanford. The *aim* now is (not to provide a better coverage but) to maximally reduce the size of the grammar, while maintaining its coverage. To off-line F-generalisation of the typed FSs in the grammar will correspond on-line specialisation of the parsing (possibly partial) results.

[1] For instance the HPSG for English developed at Stanford [5] called LinGO, two HPSGs for Japanese developed at Tokyo University [8], and respectively at DFKI–Saarbruecken, Germany [14], and the HPSG for German, developed also at DFKI [10]. Large-scale LFG grammars were developed by Xerox Corp., but they are not publicly available.

The *motivation* behind this work on F-generalisation: measurements done with LinGO on the CSLI test suite revealed that more than 50% of the feature constraints in the (expanded) grammar can be eliminated without affecting the coverage.[2] As a consequence, the unification time — the most consuming task during parsing with such grammars — will be significantly reduced, provided that the number of parse items (i.e. partial parses) does not inflate.

As a consequence, the *future tasks* will be to design more efficient algorithms for F-generalisation of LinGO-like grammars, and to examine/find techniques for keeping the number of partial parses close to that in the original grammar. The link with stochastic learning and (shallow) parsing techniques will be explored.

References

1. H. Aït-Kaci and A. Podelski. Towards a meaning of LIFE. *Journal of Logic Programming*, 16:195–234, 1993.
2. H. Aït-Kaci, A. Podelski, and S.C. Goldstein. Order-sorted feature theory unification. *Journal of Logic, Language and Information*, 30:99–124, 1997.
3. B. Carpenter. *The Logic of Typed Feature Structures - with applications to unification grammars, logic programs and constraint resolution.* Cambridge University Press, 1992.
4. L.-V. Ciortuz. Compiling HPSG into C. Unpublished technical report, The German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Germany, 2001.
5. A. Copestake, D. Flickinger, and I. Sag. *A Grammar of English in HPSG: Design and Implementations.* Stanford: CSLI Publications, 1999.
6. R. M. Kaplan and J. Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–381. The MIT Press, 1982.
7. R. Malouf, J. Carroll, and A. Copestake. Efficient feature structure operations without compilation. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):29–46, 2000.
8. Y. Mitsuishi, K. Torisawa, and J. Tsujii. HPSG-Style Underspecified Japanese Grammar with Wide Coverage. In *Proceedings of the 17th International Conference on Computational Linguistics: COLING-98*, pages 867–880, 1998.
9. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
10. Stefan Müller. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche.* Number 394 in Linguistische Arbeiten. Max Niemeyer Verlag, Tübingen, 1999.
11. C. Pollard and I. Sag. *Head-driven Phrase Structure Grammar.* Center for the Study of Language and Information, Stanford, 1994.
12. S. Shieber. *An introduction to unification-based approaches to grammars.* CSLI Publications, Stanford, 1986.
13. S. M. Shieber, H. Uszkoreit, F. C. Pereira, J. Robinson, and M. Tyson. The formalism and implementation of PATR-II. In J. Bresnan, editor, *Research on Interactive Acquisition and Use of Knowledge.* SRI International, Menlo Park, Calif., 1983.
14. M. Siegel. HPSG analysis of japanese. In *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer Verlag, 2000.
15. G. Smolka. Feature-constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.
16. K. Torisawa, K. Nishida, Y. Miyao, and J. Tsujii. An HPSG parser with CFG filtering. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG), 2000.
17. H. Uszkoreit. Categorial Unification Grammar. In *International Conference on Computational Linguistics (COLING'92)*, pages 498–504, Nancy, France, 1986.

This article was processed using the LaTeX macro package with LLNCS style

[2] This is due to the applicability of the so-called Quick-Check pre-unification filter [7].

Semi-Automated Discovery in Zariski Spaces (A Proposal)

Simon Colton, Roy McCasland, Alan Bundy and Toby Walsh,
Institute of Representation and Reasoning
Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh. EH1 1HN. UK.

We discuss the proposed application of the HR discovery system (and the various pieces of mathematical software it uses) to discovery tasks in the domain of Zariski Spaces. We propose an incremental approach using HR in a semi-automated, cross-domain fashion in domains of increasing difficulty related to Zariski spaces. This work is to be funded by an EPSRC 1-year visiting fellowship to employ Roy McCasland.

Zariski Spaces

Zariski Spaces were introduced in 1998 [MMS98]. In order to understand these spaces, one needs to first understand Zariski Topologies. To this end, let R be a commutative ring with unity, and let *specR* denote the collection of prime ideals of R. Now for each (possibly empty) subset A of R, let the *variety* of A be given by: $V(A) = \{P \in specR : A \subseteq P\}$. It is easily shown that the collection of all such varieties constitutes (the closed sets of) a topology, called the Zariski Topology on R, which we denote by $\zeta(R)$. It turns out that every topology is a semiring, if one takes the operations of addition and multiplication to be set-theoretic intersection and union respectively. Now let M be an R-module, and repeat the above process. That is to say, let *specM* denote the collection of all prime submodules of M, and for each subset B of M, let the variety of B be given by: $V(B) = \{P \in specM : B \subseteq P\}$. Finally, let $\zeta(M)$ represent the collection of all varieties of subsets of M. Then one can show that while $\zeta(M)$ seldom forms a topology, it does form a semimodule over the semiring $\zeta(R)$, where the operation in $\zeta(M)$ is taken to be intersection, and scalar multiplication is given by $V(A)V(B) = V(RAB)$.

There are reasons to suppose that Zariski Spaces might turn out to be of considerable importance in mathematics. For instance, Zariski Topologies and the study of varieties have played an enormous role in the development of Algebraic Geometry, and in particular, the Hilbert Nullstellensatz, which is one of the fundamental results in Algebraic Geometry. It is certainly possible that Zariski Spaces could have a similar impact on some branch of mathematics. Furthermore, some preliminary results suggest a possible connection between a certain concept in Zariski Spaces, called subtractive bases, and direct sum decompositions within a large class of modules. The search for direct sum decompositions has been a major undertaking in mathematics for some time, and has so far proven quite intractable, except in special cases. The study of semimodules in general has already yielded many applications to computer science [Gol92], and since Zariski Spaces are first and foremost semimodules, it is possible that their study will promote further advances in theoretical computer science.

Proposed Discovery Methods

The HR program [Col00] has been successful in making discoveries in number theory [Col99] and algebraic domains [CM01]. HR is comprised of four modules which generate four types of information, namely objects of interest, concepts which classify those objects, conjectures which relate the concepts and proofs which explain the conjectures. HR calls third party software to achieve various tasks, including computer algebra systems, constraint solvers and model generators to generate objects of interest (for both exploration and counterexamples) and theorem provers to prove conjectures. At present, HR is autonomous, i.e., the user sets some parameters for the search it will perform, then HR builds a theory and the user employs various tools to extract information about the theory. We propose to extend the theory of theory formation upon which HR is based by enabling any of the four modules to be replaced on occasion by human intervention. That is, we intend to make HR semi-automated by allowing the user to provide proofs and counterexamples to conjectures HR makes and to specify related concepts and conjectures to base the theory formation around. Alongside the development of HR's functionality, we also intend to develop the application to Zariski spaces. Zariski spaces represent a higher level of complexity than the domains in which HR has so far been applied, and the new application will require an incremental approach whereby (i) HR is enabled to form theories about increasingly complicated domains related to Zariski spaces (ii) testing is performed to see if HR invents various concepts and conjectures required for it to proceed and (iii) theory formation is centred around the important concepts and the results analysed for any discoveries. The proposed route to Zariski spaces is via: semigroups, semirings and semimodules, followed by groups, rings and modules and finally, using a cross domain approach, Zariski spaces.

It is unclear at the moment how HR will interact with the user and with third party pieces of mathematical software on this project. It seems likely that HR will be used for an initial exploration of each domain, to: (a) invent core concepts (b) prove some fundamental theorems using a theorem prover (HR has access to Otter, E, Spass and Bliksem through the MathWeb Software Bus), and (c) generate some examples of the concepts using a model generator or computer algebra system (HR uses MACE and Maple, also possible through MathWeb). Following the initial investigation, the user will both prune uninteresting concepts and specify which concepts should be emphasised in the next theory formation session. The user will be more involved in that session, choosing to prove theorems, provide counterexamples and direct the search where appropriate. By improving HR to enable such an interaction, we hope this approach will lead to the discovery of new results about Zariski spaces.

References

[CM01] S Colton and I Miguel. Constraint generation via automated theory formation. In *Proceedings of CP-2001*, pages 575–579, 2001.

[Col99] S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences*, 2, 1999.

[Col00] S Colton. *Automated Theory Formation in Pure Mathematics.* PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 2000.

[Gol92] J Golan. *The theory of Semirings with Applications in Mathematics and Theoretical Computer Science.* John Wiley and Sons, 1992.

[MMS98] R McCasland, M Moore, and P Smith. An introduction to Zariski spaces over Zariski topologies. *Rocky Mountain Journal of Mathematics*, 28:1357–1369, 1998.

Planning the Whisky Problem

Louise A. Dennis[1] and Alan Bundy[2]

[1] School of Computer Science and Information Technology, University of Nottingham, Nottingham, NG8 1BB lad@cs.nott.ac.uk
[2] Division of Informatics, University of Edinburgh, 80 South Bridge, Edinburgh, EH1 1HN bundy@dai.ed.ac.uk

Proof critics are a technology from the proof planning paradigm. They examine failed proof attempts in order to extract information that can be used to generate a patch which will allow the proof to go through.

We examine the proof of the "whisky problem", a challenge problem from the domain of temporal logic. The proof requires a generalisation of the original conjecture and we examine two proof critics which can be used to create this generalisation. Using these critics we believe we have produced the first fully automatic proofs of this challenge problem.

We chose to use the proof planning system $\lambda Clam$ [2] to attempt an automation of the proof of the whisky problem. The whisky problem is represented in $\lambda Clam$ as three axioms and a goal to be proved. The axioms are:

$$\begin{aligned} & p(a,0), \\ \forall x.\, p(x,0) \quad & \Rightarrow \quad p(h(x),0), \\ \forall x,y.\, p(h(x),y) \quad & \Rightarrow \quad p(x,s(y)). \end{aligned}$$

These are treated as the following rewrite rules in $\lambda Clam$:

$$p(a,0), \tag{1}$$

$$p(h(X),0) \quad :\Rightarrow \quad p(X,0), \tag{2}$$

$$p(X,s(Y)) \quad :\Rightarrow \quad p(h(X),Y). \tag{3}$$

The goal to be proved is

$$\forall x.\, p(a,x).$$

The challenge is that for a proof to go through the goal must be generalised to $\forall x,n.\, p(h^n(a),x)$ where h^n signifies n applications of h. A $\lambda Clam$ inductive proof of this fails to apply the rewrite rule (3) in the induction to the goal $p(a,n) \vdash p(a,s(n))$ because the resulting embedding is unsinkable. The embedding is part of the rippling heuristic ([3]) used to control rewriting during induction. This can be used to trigger a critic.

The existing $\lambda Clam$ critic for this situation, based on work in [1], generalises the initial goal to

$$\forall x,y.\, p(F(y),x).$$

Where F is gradually instantiated during the course of the proof to the desired function. This proof proved sensitive to the rewrite rules available in the default

context and required the provision of a new induction scheme:

$$\frac{P(B) \quad P(G^n(B)) \vdash P(G(G^n(B)))}{\forall n.P((\lambda x.G^x(B))(n))}, \tag{6}$$

We also investigated a more specialised critic based on work in [4] which speculated the "target" generalisation of

$$\forall x, n.\ p(h^n(a), x).$$

directly based on the contents of the wave front in the unsinkable embedding. The proof then went through in the default $\lambda Clam$ context.

As far as we are aware these represent the first fully automatic proofs of the whisky problem.

Acknowledgments

This research was funded by EPSRC grants Gr/M45030 and Gr/M46624. We thank the members of the Logics Group at Liverpool University and Michael Fisher and Anatoli Degtyarev in particular for drawing our attention to this challenge problem.

References

1. A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996. Also available as DAI Research Paper No 716, Dept. of Artificial Intelligence, Edinburgh.
2. J.D.C. Richardson, A. Smaill, and I. Green. System description: Proof planning in higher-order logic with lambda-clam. In C. Kirchner and H. Kirchner, editors, *Conference on Automated Deduction (CADE'98)*, volume 1421 of *Lecture Notes in Computer Science*, pages 129–133. Springer-Verlag, 1998.
3. A. Smaill and I. Green. Higher-order annotated terms for proof search. In J. von Wright, J. Grundy, and J. Harrison, editors, *Theorem Proving in Higher Order Logics: 9th International Conference, TPHOLs'96*, volume 1275 of *Lecture Notes in Computer Science*, pages 399–414, Turku, Finland, 1996. Springer-Verlag. Also available from Edinburgh as DAI Research Paper 799.
4. T. Walsh. A divergence critic for inductive proof. *Journal of Artificial Intelligence Research*, 4:209–235, 1996.

The Relationship Between Temporal Logic Normal Forms and Alternating Automata

Clare Dixon[†], Alexander Bolotov[‡] and Michael Fisher[†]

[†] Department of Computer Science, University of Liverpool, Liverpool L69 7ZF,
{M.Fisher,C.Dixon}@csc.liv.ac.uk

[†] Harrow School of Computer Science, University of Westminster, Harrow HA1 3TP
A.Bolotov@westminster.ac.uk

Introduction

The connection between temporal logic and automata of different kinds is well known. A model for a propositional linear-time temporal logic formula, φ, is essentially a sequence of states where the propositions from φ are set to true or false such that the sequence of states and setting of propositions satisfies φ. This sequence can be viewed as an infinite word over subsets of the propositions in φ. Thus for any propositional linear-time temporal logic formula we can construct a finite automaton such that the automaton accepts exactly the sequence of states (infinite word) which satisfies the formula [7]. If the propositional linear-time temporal logic formula is unsatisfiable then the automata constructed is empty.

SNF (Separated Normal Form) is a normal form for representing propositional linear-time temporal logic (PLTL) formulae [6]. The normal form comprises formulae that are implications with present-time formulae on the left-hand side and (present or) future-time formulae on the right-hand side. The transformation into the normal form reduces most of the temporal operators to a core set and rewrites formulae to be in a particular form. The transformation into SNF depends on three main operations: the renaming of complex subformulae; the removal of temporal operators; and classical style rewrite operations. SNF has been used as the basis for a temporal resolution method [6] and for execution of temporal formulae [1]. In previous work we have considered the relationship between Büchi Automata and SNF [2].

An alternating automata is an automata that has both the power of existential and universal choice, similar in structure to an and/or graph. Alternating automata were considered in [3, 4] and have been studied in relation to temporal logic in [7] for example.

Alternating Automata

We define alternating automata following [7]. A is an alternating automata $A = (\Sigma, S, s_0, \rho, F)$ such that

- Σ is a finite non-empty alphabet;
- S is a set of states;
- $s_0 \in S$ is an initial state;
- $\rho : S \times \mathcal{P}(\Sigma) \rightarrow \mathcal{B}^+(S)$ is a transition function

- $F \subseteq S$ is a set of accepting states;

where $\mathcal{B}^+(S)$ are the set of positive Boolean formulae over S, i.e. Boolean formulae built from S using $\wedge$ and $\vee$. Runs in alternating automata are trees. In the following, given a tree τ, ϵ is the root node of τ, x is a node in τ and $|x|$ is the distance of x from the root, where $|\epsilon| = 0$. An W-labelled tree, for some set W is a tree, τ, and mapping t such that t maps nodes of τ to W. A *run* of A on an infinite word $w = a_0, a_1, \ldots$ is a possibly infinite S-labelled tree r such that $r(\epsilon) = s_0$ and the following holds. If $|x| = i$, $r(x) = s$ and $\rho(s, a_i) = \theta$, then x has children $x_1, \ldots x_k$ for some $k \leq |S|$ and $\{r(x_1), \ldots, r(x_k)\}$ satisfies θ. The run is *accepting* if every infinite branch in r includes infinitely many labels in F. Note the run can also have finite branches; if $|x| = i$, $r(x) = s$ and $\rho(s, a_i) =$ **true** then x does not need to have any children. However we cannot have $\rho(s, a_i) =$ **false** as **false** is not satisfiable. Thus every branch in an accepting run has to hit **true** or hit accepting states infinitely often.

The Translation Between SNF and Alternating Automata

We provide transformations from alternating automata into SNF and from SNF into alternating automata in [5]. We show that a set, R, of SNF clauses is satisfiable if and only if the alternating automata, A_R, constructed from R has an accepting run. Similarly an alternating automata A has an accepting run if and only if the set of clauses R_A constructed from A is satisfiable.

This is part of ongoing work looking at the relationship between SNF and other formalisms. We were prompted to investigate the connection between SNF and alternating automata due to and/or structure of SNF i.e. SNF is a conjunction of clauses which are essentially disjunctions. In particular, this result is useful as there is no direct method for checking the non-emptiness of an alternating automata. However non-emptiness can be checked by, for example, giving a translation into a (nondeterministic) Büchi automata (see for example [7]) and doing the non-emptiness check there. Here, we can use the translation provided in [5] and then apply temporal resolution to the clauses obtained. If the clauses obtained are unsatisfiable then this means there is no accepting run.

References

[1] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logic.* Research Studies Press, May 1996.

[2] A. Bolotov, M. Fisher, and C. Dixon. On the Relationship between w-Automata and Temporal Logic Normal Forms. *Journal of Logic and Computation*, 2002. To appear.

[3] J. Brzozowski and E. Leiss. Finite automata, and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[4] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *ACM Journal*, 28(1):114–133, 1981.

[5] C. Dixon, A. Bolotov, and M. Fisher. Alternating Automata and Temporal Logic Normal Forms. In preparation.

[6] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, January 2001.

[7] M. Y. Vardi. An Automata-Theoretic Approach to Linear Temporal Logic. In *Proceedings of Banff'94*, Lecture Notes in Computer Science, pages 238–226. Springer-Verlag, 1994.

Comparing SAT preprocessing techniques

Lyndon Drake* lyndon@cs.york.ac.uk
Alan Frisch* frisch@cs.york.ac.uk
Inês Lynce† ines@sat.inesc.pt
João Marques-Silva† jpms@sat.inesc.pt
Toby Walsh* tw@cs.york.ac.uk

28 February 2002

1 Introduction

Preprocessing algorithms can dramatically improve the performance of even the most efficient SAT solvers on many problem instances [8]. Such algorithms simplify the formula by various means, including the deduction of necessary assignments, the addition of implied clauses, the deletion of redundant clauses, and the identification of equivalent literals. A number of preprocessors have been made publicly available, including `Compact` [2], `Compactor` [6], `Simplify` [9], and `CompressLite` [4]. While each preprocessor is very successful on some instances, they each combine a different selection of simplification techniques, making it difficult to understand which techniques are responsible for their success on a particular problem.

We are in the process of implementing each of the main simplification techniques in JQuest, a Java framework designed to support comparisons of SAT techniques. JQuest has already been used to evaluate the performance of various data structures for SAT solvers [7]. In addition to comparing existing techniques, we are creating efficient implementations of those techniques and investigating novel techniques. Here we briefly discuss two existing techniques: length-restricted resolution and binary equivalence finding.

2 Length-restricted resolution

One common technique, used for example in `Compactor` [6], is to do all possible binary resolutions where the resolvent has at most three literals. Other variations are possible, including limiting the size of the resolvents to two literals, and only doing resolutions where both parents will be subsumed. We also intend to investigate the effect on the branching heuristic of adding resolvents to the formula.

In practice, one problem with resolution is that resolvents are often subsumed by existing clauses in the formula. Adding redundant clauses is unlikely to make the instance simpler to solve, and so should be avoided if possible. We have a fast method for subsumption checking of binary and ternary clauses based on hash tables, which can be used to efficiently identify redundant resolvent clauses. We also have a similar technique, as yet unimplemented, for identifying and deleting clauses in the original formula if they are subsumed by a resolvent clause.

3 Binary equivalence finding

Some SAT instances, particularly those mapped from real world problems, contain equivalent literals [5]. For example, if a formula contains the clauses $(\neg a \vee b)$ and $(a \vee \neg b)$, we know that a and b

[1]University of York, Heslington, York YO10 5DD, United Kingdom. This work and the first author are supported by EPSRC Grant GR/N16129 (see http://www.cs.york.ac.uk/aig/projects/implied).
[2]Technical University of Lisbon, IST/INESC/CEL, R. Alves Redol, 9, 1000-029 Lisbon, Portugal

are equivalent because a implies b and b implies a. Loops of binary equivalences are possible, such as $(\neg a \vee b)$, $(\neg b \vee c)$, and $(a \vee \neg c)$, in which a, b, and c are all equivalent. Given a set of equivalent literals, one of the literals in the set can replace all occurrences in the formula of the other literals in the set, without altering the satisfiability of the formula. This replacement reduces the number of variables in the formula (potentially reducing the number of assignments during the search), and can also identify tautological clauses (which can safely be removed from the formula).

The implications represented by binary clauses can be represented in a graph, in which sets of equivalent literals will appear as strongly connected components (SCCs). Tarjan [10] gives a linear time algorithm for finding SCCs in a graph, applied to SAT by Aspwall et al [1] and later de Val [3]. Because Tarjan's SCC algorithm is linear, binary equivalence finding is an particularly cost-effective technique.

4 Future work

A number of techniques still need to be implemented, in particular those that deduce necessary assignments. Once our implementation work has been completed, we intend to systematically compare the performance of the various preprocessing techniques. We also plan to theoretically study the relationships between the techniques.

5 Summary

Preprocessing techniques for SAT have not previously been experimentally compared in a systematic fashion. We are in the process of implementing many of the published techniques in a common platform (JQuest) in order to compare their performance.

References

[1] Aspwall, Plass, and Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, pages 121–123, March 1979.

[2] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 21–27, 1993.

[3] Alvaro de Val. Simplifying binary propositional theories into connected components twice as fast. In *LPAR'2001, 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science, pages 389–403. Springer-Verlag, 2001.

[4] Daniel le Berre. Exploiting the real power of unit propagation lookahead. In *LICS Workshop on Theory and Applications of Satisfiability Testing*, June 2001.

[5] Chu Min Li. Integrating equivalency reasoning into Davis-Putnam procedure. In *Proceedings of AAAI-2000*, pages 291–296, Austin, Texas, USA, July 2000.

[6] Chu Min Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 1997.

[7] I. Lynce and J. Marques-Silva. Efficient data structures for fast sat solvers. Technical report, Instituto de Engenharia de Sistemas e Computadores, November 2001.

[8] Inês Lynce and João P. Marques-Silva. The interaction between simplification and search in propositional satisfiability. In *CP2001 Workshop on Modelling and Problem Formulation (Formul'01)*, Paphos, Cyprus, December 2001.

[9] João P. Marques-Silva. Algebraic simplification techniques for propositional satisfiability. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 537–542, September 2000.

[10] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.

Adding a Zoom to Linear Temporal Logic*

Ulrich Endriss

Department of Computer Science, King's College London
Strand, London WC2R 2LS, United Kingdom
Email: endriss@dcs.kcl.ac.uk

1 Introduction

Temporal logic. One of the great success stories of non-classical logics in mainstream computer science is that of linear temporal logic and its applications to systems specification and verification [2]. The simple formalism of a sequence of states and a valuation function mapping atomic propositions to sets of states, combined with propositional and modal operators to construct complex formulas, is surprisingly powerful when describing the behaviour of a given system over time.

However, this logic does *not* support the concept of modularisation in a natural manner. Each time we want to add additional detail about a subsystem associated with one of the states, we have to revise the specification of the entire system.

Zooming in. We propose to remedy this shortcoming *by adding a zoom to linear temporal logic:* If we (recursively) relate every state with another time line to describe the behaviour of the subsystem associated with that state we obtain a tree-like structure. In fact, we obtain an *ordered tree*: the relation connecting a state with all the states in the time line beneath it is a tree (if we also introduce a root) and the children of each node (the states of a single time line) are ordered. In this paper, we discuss a modal logic with frames that are ordered trees. It provides modal operators working both along the branches of a tree (level-of-detail dimension) and along the order declared over the children of a node (temporal dimension).

We define syntax and semantics of this new logic in Section 2 and briefly sketch a decidability proof in Section 3. A full paper is currently in preparation [1].

2 Ordered Tree Logic

We present syntax and semantics of the modal logic of (discretely) ordered trees.

Syntax. The set of *well-formed formulas* A of our logic is formed according to the following BNF production rule (p stands for propositional letters):

$$A \;::=\; p \mid \neg A \mid A \wedge A \mid \ominus A \mid \Theta A \mid \bigcirc A \mid \\ \Diamond\!\!\!\!-\, A \mid \Diamond\!\!\!\!\cdot\, A \mid \Diamond\!\!\!\!\!-\, A \mid \Diamond\!\!\!\!\cdot\, A \mid \Diamond\!\!\!\!-^{+} A$$

Additional propositional connectives and box-operators may be introduced as defined operators in the usual way (like, for example, $\boxminus\varphi = \neg\Diamond\!\!\!\!-\neg\varphi$).

Semantics. A *discretely ordered tree* $\mathcal{T}$ is a tree where the children of each node form a *discrete order* (that is, between any two sibling nodes there can only be finitely many other nodes). A *model* is a pair $\mathcal{M} = (\mathcal{T}, V)$, where $\mathcal{T}$ is such an ordered tree and V is a valuation function mapping propositional letters to sets of nodes of $\mathcal{T}$. Truth of a formula φ in $\mathcal{M}$ at a node $t \in \mathcal{T}$ is defined as follows:

1. $\mathcal{M}, t \models p$ iff $t \in V(p)$ for propositional letters p
2. $\mathcal{M}, t \models \neg\varphi$ iff not $\mathcal{M}, t \models \varphi$
3. $\mathcal{M}, t \models \varphi \wedge \psi$ iff $\mathcal{M}, t \models \varphi$ and $\mathcal{M}, t \models \psi$
4. $\mathcal{M}, t \models \bigcirc\varphi$ iff t has a parent t' and $\mathcal{M}, t' \models \varphi$
5. $\mathcal{M}, t \models \ominus\varphi$ iff t has a left neighbour t' and $\mathcal{M}, t' \models \varphi$
6. $\mathcal{M}, t \models \Theta\varphi$ iff t has a right neighb. t' and $\mathcal{M}, t' \models \varphi$
7. $\mathcal{M}, t \models \Diamond\!\!\!\!-\varphi$ iff t has an ancestor t' with $\mathcal{M}, t' \models \varphi$
8. $\mathcal{M}, t \models \Diamond\!\!\!\!\cdot\varphi$ iff t has a left sibling t' with $\mathcal{M}, t' \models \varphi$
9. $\mathcal{M}, t \models \Diamond\!\!\!\!\cdot\varphi$ iff t has a right sibling t' with $\mathcal{M}, t' \models \varphi$
10. $\mathcal{M}, t \models \Diamond\!\!\!\!-\varphi$ iff t has a child t' with $\mathcal{M}, t' \models \varphi$
11. $\mathcal{M}, t \models \Diamond\!\!\!\!-^{+}\varphi$ iff t has a descendant t' with $\mathcal{M}, t' \models \varphi$

A formula φ is called *satisfiable* iff it has a model (i.e. iff there are $\mathcal{M} = (\mathcal{T}, V)$ and $t \in \mathcal{T}$ with $\mathcal{M}, t \models \varphi$).

3 Bounded Finite Models

Ordered tree logics can be shown to be decidable by establishing a bounded finite model property (fmp), that is, by showing that any formula φ that is satisfiable in *some* model is also satisfiable in a model of limited size (where the maximal size is a function of the length of φ). Our techniques are similar to those used by Sistla and Clarke [3] to establish upper complexity bounds for propositional linear temporal logics.

Theorem 1 (Bounded FMP) *The modal logic of discretely ordered trees has the bounded finite model property.*

Theorem 1 is a corollary to Lemmas 2 and 5, proofs for which we are going to informally sketch below. The following observation will play a central role: To check whether a given formula φ is true at some node in a given model we have to check whether certain subformulas of φ are true at certain (other) nodes in the model; formulas that are not subformulas of φ are not relevant. So instead of models, we can work with *type models*, that is ordered trees where each node is associated with a certain *type* (a set of subformulas of the input formula φ).

Bounded branching. We first work towards controlling the horizontal dimension and show how we can reduce the branching factor of a given model.

Lemma 1 (Sibling pruning) *The stretch between two siblings of the same type can be removed (including one of the end nodes) without affecting satisfiability, provided we keep the node of evaluation as well as a witness for each formula of the form $\Diamond\!\!\!\!\cdot\varphi$ or $\Diamond\!\!\!\!\cdot^{+}\varphi$ in the parent.*

*This work has been supported by the EPSRC under grant reference numbers GR/R45369 and GR/N23028.

We can show this by structural induction. For atoms and propositional connectives, pruning elsewhere in the tree does not affect satisfiability. Formulas of the form $\Diamond\varphi$ or $\Diamond^+\varphi$ are not affected by definition. Neither are formulas of the form $\bigcirc\varphi$ or $\Diamond\varphi$ as they are either removed from the tree or refer to nodes unaffected by the pruning operation. The horizontal modalities provide the interesting cases. We exemplify the general idea for formulas of the form $\Diamond\varphi$. Suppose $\Diamond\varphi$ is true at a node somewhere to the left of the stretch to be removed (we need to check that at least one witness survives). If $\Diamond\varphi$ is also true at the left one of the 'pruning nodes', then it must equally hold at the right one (because they have the same type), i.e. φ is true somewhere to the right of the stretch (and we are done). Otherwise, φ must already hold somewhere before the stretch (and we are done as well).

We can use the Sibling Pruning Lemma to prove the following result:

Lemma 2 (Bounded branching) *A formula φ of length n is satisfiable iff it is satisfiable in a (periodic) type model with a maximal branching factor of $(n+1)\cdot 2^n$.*

Observe that for any set of children there are at most n nodes the Sibling Pruning Lemma has to respect (the node of evaluation and up to $n-1$ witnesses), but it can be freely applied in between those n nodes. There are up to n subformulas of φ and, hence, up to 2^n consistent types, so we can reduce the $n-1$ finite stretches in between to at most 2^n-2 nodes each.

The left- and rightmost stretch, however, may be infinite. This is where periodicity comes into play. We consider the rightmost stretch: In case it is finite, we can reduce it to 2^n-1 nodes. Otherwise, there is a point from which onward all types that *do* come up come up infinitely often. Each one of them must have another type that occurs infinitely often as its right-hand neighbour. Hence, we can apply the Sibling Pruning Lemma in such a way that we obtain a periodic stretch of types (by always pruning between a node and the next occurrence of a node of the same type together with its 'designated neighbour'). The rightmost stretch as a whole (including the period) can then be pruned down to 2^n-1 nodes.

Altogether, we get a maximum of $n+(n-1)\cdot(2^n-2)+2\cdot(2^n-1) < (n+1)\cdot 2^n$ nodes.

Bounded depth. Now we turn to the vertical dimension and show how to reduce the length of branches in a given tree, again, without affecting satisfiability.

Lemma 3 (Branch pruning) *If a node t_1 and its descendant t_2 have the same type, we can replace the subtree beneath t_1 with the tree beneath t_2 without affecting satisfiability, provided we do not remove the node of evaluation.*

We omit the proof, which is similar to that of the Sibling Pruning Lemma.

To allow for the finite representation of recursive trees we introduce the notion of a *link*: a tree with a link from a node t_1 to another node t_2 represents the tree we get by (recursively) replaceing t_1 with t_2 (together with the subtrees beneath them).

Lemma 4 (Link introduction) *If two nodes t_1 and t_2 have the same type, we can introduce a link from t_1 to t_2 without affecting satisfiability, provided we keep the node of evaluation and a witness for each formula of the form $\Diamond^+\varphi$ in t_2.*

Again, we omit the proof and only point out that the ideas are essentially the same as before.

Lemma 5 (Bounded depth) *A formula φ of length n is satisfiable iff it is satisfiable in a type model (with links) with a maximal branch length of 2^{n+1}.*

Let us first observe that we can use the Branch Pruning Lemma to ensure that every type has a first occurrence at a node of depth $\leq 2^n$ (there are up to 2^n types and we can shorten any branch that has more than one node of the same type). We pick a minimal 'upper part' of the tree that features each type at least once.

Then we turn all branches into finite branches by applying the Link Introduction Lemma in such a way that links point from a node in the 'lower part' to one in the 'upper part'. We can always find a node far enough down the tree so that the lemma becomes applicable (i.e. that we keep all the required witnesses).

Finally, we use again the Branch Pruning Lemma, this time to reduce the 'lower part' to a maximal height of 2^n. Observe that, as we restrict pruning to the 'lower part' alone, no 'link-heads' will be cut off.

Decidability. Decidability is a direct consequence of the bounded fmp: a naïve decision procedure could simply enumerate all potential models up to the known maximal size and check each one of them. Hence, we obtain the following main result:

Theorem 2 (Decidability) *The satisfiability problem for the modal logic of discretely ordered trees is decidable.*

4 Conclusion

We conclude with a brief outlook on possible directions for future research in this area.

Complexity analysis. A bounded fmp provides a first step towards a complexity analysis of the logic under investigation. However, given that our bounds are exponential in *both* dimensions of a tree, at this stage, we can only establish a NEXPTIME upper bound and it is not clear if (and how) this could be improved upon.

Extensions. A number of interesting extensions to our logic, both to the language and to the underlying semantics, are possible. One such extension would be to investigate the addition of the temporal operators *since* and *until* (certainly to the horizontal dimension, but possibly also along branches). On the semantical level, an interesting extension would be to drop the condition of discreteness for the order declared over sibling nodes and to consider dense orders or general linear orders.

References

[1] U. Endriss. A Modal Logic of Ordered Trees. Manuscript, 2002.

[2] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[3] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, 32(3):733–749, 1985.

Automatically Transforming Constraint Satisfaction Problems: Further Progress

Alan M. Frisch, Ian Miguel
AI Group
Dept. Computer Science
University of York, York, England
{frisch,ianm}@cs.york.ac.uk

Toby Walsh
Cork Constraint Computation Center
University College Cork
Ireland
tw@4c.ucc.ie

1 Introduction

A constraint satisfaction problem (CSP) consists of a triple, (X, D, C), where X is a set of variables, D is a set of corresponding domains of values and C is a set of constraints which specify allowed combinations of assignments of values to variables. Constructing an effective model of a CSP is, however, a challenging task. Carefully chosen transformations of a basic model can greatly reduce the amount of effort required to solve the problem by systematic search (e.g. [5]). These include adding constraints that are implied by other constraints in the problem, adding constraints that remove symmetrical solutions to the problem, removing redundant constraints and replacing constraints with their logical equivalents.

1.1 CGRASS

Outwith a highly focused domain like planning [2], model transformation is typically performed in an ad-hoc manner by hand. We describe the CGRASS (Constraint GenERation And Symmetry-breaking) system that can improve a problem model by automatically performing appropriate transformations. The system architecture is based on, and extends, Bundy's 'proof planning' [1].

Proof planning is used to guide the search for a proof in automated theorem proving. Common patterns in proofs are identified and encapsulated in *methods*. A proof planner takes a goal to prove, and selects from a database of methods one which matches this goal. The proof planner checks that the pre-conditions of the method hold. If so the proof planner executes the post-conditions. This constructs the output goal or goals.

Strong method preconditions limit transformations to those that are likely to produce a simplified problem. Methods act at a high level, performing complex transformations that might require complex proofs to justify at the individual inference rule level. Search control is cleanly separated from the inference steps. We can therefore try out a variety of search strategies.

CGRASS' input currently consists of a statement of the initial problem variables and their domains and a list of constraints. We intend to enable CGRASS to accept the OPL language [6] as input, as well as specialised extensions such as the ability to support matrices of variables directly. Output is created in the same simple language. Hence, very little effort is necessary for translation into the required input for a variety of existing solvers.

Fundamental to the efficiency of CGRASS is the normal form it imposes on the constraint set, inspired by that used in the HARTMATH computer algebra system[1]. A normal form allows us, to an extent, to replace the test for semantic equivalence with a simpler syntactic comparison. The normal form used combines a lexicographic ordering and simplifications such as collection of like terms, cancellation and removal of common factors.

2 Methods

The following set of methods is not complete in any sense, and a principle item of future work is its extension.

Symmetry Often the most useful constraints can only be derived when some or all symmetry has been broken. Hence, CGRASS attempts to detect symmetry as a pre-processing step. Symmetrical variables have identical domains and are such that, if all occurrences of the pair in the constraint set are exchanged and the constraint set is re-normalised it returns to its original state. A similar process is used to check for symmetrical sub-terms. Symmetry is broken by creating a partial order between symmetrical variables/sub-terms.

Introduce This method binds a new variable to a non-atomic sub-term that recurs in the constraint set. Variable introduction is a powerful tool which, in combination with `eliminate` (below), can tighten the constraint graph and reduce constraint arity. Tightening the constraint graph means that propagation on the introduced variable's domain has a wider reaching effect. Reducing a constraint's arity means that fewer variables need to be instantiated before it can be used to prune the search space.

Eliminate This method performs Gaussian-like variable elimination. The preconditions identify variables or terms which can be eliminated from a more complex constraint, insisting that the resulting constraint has a smaller size (in terms of number of constituent terms) than the original.

[1] `http://www.hartmath.org`

Other Methods The `genAllDiff` method greedily generates an all-different constraint (for which we have powerful pruning methods [4]) from a clique of inequations. Other simple, but useful, methods also exist. For example, `nodeConsistency` prunes domain elements given an algebraic constraint consisting of a variable and a constant. These methods are not only cheap to fire, but often result in a reduction in the size of the constraint set. This promotes efficiency by leaving fewer constraints for the more complicated methods to attempt to match against.

3 Issues

In constructing CGRASS we have extended proof planning along a number of dimensions:

Non-monotonicity: CGRASS' methods sometimes add a new constraint, at others replace a constraint by a tighter one, or eliminate a redundant constraint. The constraint set may therefore increase or decrease. Hence, we replace the method 'output' by 'add' and 'delete' lists as used in classical planning.

Pattern matching: Existing proof planners typically use first order unification to match a method's input against the current goal or subgoal. CGRASS uses a richer pattern matching language specialised to the task of reasoning about sets of constraints.

Looping: Unless a method deletes some of its input constraints, its preconditions continue to hold, allowing repeated firing. We incorporated an history mechanism into CGRASS to prevent this.

Termination: Previous applications of proof planning have a clear termination condition: goals are reduced to subgoals until all are proven. In transforming CSPs it is much less clear. We must decide when to stop making transformations and search for an answer. CGRASS' methods currently have strong enough preconditions that they can be run to exhaustion. We may in the future have to add a proof critic-style executive [3] which terminates proof planning when future rewards look poor.

Constraint utility: CGRASS uses measures like constraint arity and tightness to eliminate obviously useless constraints. We are inventing heuristics to help with the difficult decision as to which of the remaining derived constraints to keep.

Explanation: In order for the user to see how a new model was derived, we adapted the existing proof planning tactic mechanism. CGRASS' tactics write out text explaining the application of the methods.

4 The Golomb Ruler Problem

We illustrate the operation of CGRASS by means of the Golomb ruler problem [5]. A Golomb ruler is a set of n ticks at integer points on a ruler of length m such that all the inter-tick distances are unique. A concise model:

$$\text{minimise:}\max_i(x_i)$$

$$\forall i,j,k,l\ (i \neq j) \wedge (k \neq l) \wedge (i \neq k \vee j \neq l) \rightarrow$$
$$(x_i - x_j \neq x_k - x_l) \text{ is added to the problem}$$

Taken literally, this a poor model. The constraints are quaternary, and will be delayed by most solvers. There is also a large amount of symmetry present.

We focus on the 3-tick ruler. The basic model produces 30 constraints. CGRASS' initial normalisation of the constraint set reduces this to just 12. Symmetry testing reveals that x_1, x_2 and x_3 are symmetrical, hence $x_1 \leq x_2$ and $x_2 \leq x_3$ are added. Three variables, $z_0 = x_1 - x_2$, $z_1 = x_2 - x_3$ and $z_2 = x_3 - x_1$ are `introduced` and `eliminate` substitutes them into the quaternary constraints. Finally, `genAllDiff` fires which, coupled with the firing of some of the minor auxiliary methods, gives the concise model of table 1.

$x_1 \in \{0..7\}$	$x_2 \in \{1..8\}$	$x_3 \in \{2..9\}$
$x_1 < x_2$	$x_2 < x_3$	$x_1 \neq x_3$
$z_0 = x_1 - x_2$	$z_1 = x_2 - x_3$	$z_2 = x_3 - x_1$
All-different(z_0, z_1, z_2)		

Table 1: Golomb Ruler: Transformed Model.

This model is far easier to solve than the original. Similar models are obtained for larger Golomb rulers, leading to substantial reductions in search effort. However, larger problems generate a greater volume of input constraints via the naive input model. This creates a lot of work for CGRASS and an overall reduction in efficiency.

5 Next Steps

In order to compensate, we are extending CGRASS to support quantified expressions directly. This would immediately reduce the size of the input to two constraints in the Golomb ruler example. It would also allow us to reason about a whole *class* of problems, rather than at the level of single instances. Writing methods would typically be complicated by the need to support quantified constraints. However, certain operations would be made much easier. One example is the `genAllDiff` method where the input would be simply: $\forall i,j\ i \neq j \rightarrow x_i \neq x_j$.

Acknowledgements

This project and the second author are supported by EPSRC Grant GR/N16129[2].

References

[1] A. Bundy. A science of reasoning. In J-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–198. MIT Press, 1991.

[2] M.D. Ernst, T.D. Millstein, and D.S. Weld. Automatic SAT-compilation of planning problems. In *Proc. 15th IJCAI*, pages 1169–1176, 1997.

[3] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proof. In *Proceedings of LPAR'92, LNAI 624*. Springer-Verlag, 1992.

[4] J.C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. 12th AAAI*, pages 362–367, 1994.

[5] B.M. Smith, K. Stergiou, and T. Walsh. Modelling the golomb ruler problem. In *Proceedings of the IJCAI-99 Workshop on Non-Binary Constraints*. International Joint Conference on Artificial Intelligence, 1999.

[6] P. van Hentenryck. *The OPL Optimization Programming Language*. The MIT Press, 1999.

[2] `http://www.cs.york.ac.uk/aig/projects/implied/index.ht`

Algorithms for Guiding Clausal Temporal Resolution

M.Carmen Fernández Gago,
Logic and Computation Group
Department of Computer Science,
University of Liverpool,
Peach Stret, L69 7ZF, United Kingdom
Email: M.C.Gago@csc.liv.ac.uk

1 Introduction

Temporal logic is a variety of non-classical logic used in a range of areas within Computer Science and Artificial Intelligence, for example the verification of reactive systems [8], the implementation of temporal query languages [2], and temporal logic programming [1]. Consequently, different proof methods have been developed, implemented and applied.

In this work we use a proof method for temporal logics based upon resolution for classical logic [6]. Resolution based methods have the advantage that, as in the classical case, a wide range of strategies can potentially be used.

The temporal resolution procedure is characterised by the translation to a normal form, the application of a classical style resolution rule between formulae that occur at the same moment in time, termed *step resolution*, that is, where the clauses contain no eventualities ($\Diamond l$, i.e, *l holds now or at sometime in the future*), together with a novel temporal resolution rule, which derives contradictions over temporal sequences (*temporal resolution*). Although the clausal temporal resolution method has been defined, proved correct and implemented it sometimes generates an unneccesarily large set of formulas that may be irrelevant to the refutation. Not only that, but temporal resolution operations are applied only after all step resolution inferences have been carried out. This means that cases where a large amount of step resolution can occur the method may be very expensive.

The application of the temporal resolution rule involves searching for a set of clauses that together represent $X \Rightarrow \bigcirc \Box l$ for resolution with $\Diamond \neg l$. Rather than using an external procedure to do this, as happens in [7], we provide an algorithm to detect X systematically using only step resolution. If we need to search for a second loop we have developed an additional algorithm that can reuse the clauses generated during previous searches rather than recreating then as is necessary in [3].

2 The Search for Loops

For the application of the temporal resolution rule a loop must be detected, that is, a set of clauses $\phi_i \Rightarrow \bigcirc \psi_j$, to be resolved with $\Diamond \neg l$, such that $\bigwedge_{i=1}^{n} (\phi_i \Rightarrow \bigcirc (l \wedge \bigvee_{j=1}^{n} \phi_j))$.

Trying to find such a set of clauses we propose the following algorithm where we begin choosing a candidate for $G_i = \bigvee_{j=1}^{n} \phi_i$

1. Choose $G_{-1} \Leftrightarrow \mathbf{true}$
2. Given a guess G_i add the clause $s_i \Rightarrow \bigcirc(\neg G_i \vee \neg l)$ and apply Step Resolution.
3. For all clauses $\mathbf{true} \Rightarrow \bigcirc(\neg s_i \vee F_j)$ obtained during the proof, let $G_{i+1} \Leftrightarrow G_i \wedge (\bigvee_{j=1}^{m} \neg F_j)$.
4. Go to 2 until either

(a) $G_i \Leftrightarrow G_{i+1}$ (we terminate having found a loop).

(b) G_{i+1} is empty. (we terminate without having found a loop).

We can show

Theorem 1 (Soundness) *If the above algorithm outputs X when applied to $\Diamond\neg l$ then $X \Rightarrow \bigcirc \Box l$.*

Theorem 2 (Completeness) *If a set of clauses contains a loop X such that $X \Rightarrow \bigcirc \Box l$ then the algorithm detects X' such that $X \Rightarrow X'$.*

Proofs can be found in [5]. Essentially the proofs are based on [3].

Also in [5] we give an algorithm for searching for a second or subsequent loops that reuses clauses generated by the previous searches.

In the future we hope to apply these results to the development of strategies for temporal resolution that allows us to reduce the search space. In particular, we are interested in incorporating the set of support strategy [9, 4], to guide the search for a proof in temporal resolution, with the expectation that it will be success as in the classical case.

References

[1] M. Abadi and Z. Manna. Temporal Logic Programming. *Journal of Symbolic Computation*, 8: 277–295, 1989.

[2] J. Chomicki and D. Niwinski. On the Feasibility of Checking Temporal Integrity Constraints. *Journal of Computer and System Sciences*, 51(3):523–535, December 1995.

[3] C. Dixon. Temporal Resolution using a Breadth-First Search Algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[4] C. Dixon and M. Fisher. The Set of Support Strategy in Temporal Resolution. In *Proceedings of TIME-98 the Fifth International Workshop on Temporal Representation and Reasoning*, Sanibel Island, Florida, May 1998. IEEE Computer Society Press.

[5] M. C. Fernández-Gago, M. Fisher, and C. Dixon. Algorithms for Guiding Clausal Temporal Resolution. Current Paper. In preparation.

[6] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 99–104, Sydney, Australia, August 1991. Morgan Kaufman.

[7] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. In *Transactions on Computational Logic 2(1)*, January 2001.

[8] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[9] L. Wos, G. Robinson, and D. Carson. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *ACM Journal*, 12:536–541, October 1965.

On the relationship between decidable fragments, non-classical logics, and description logics

Lilia Georgieva[1], Ullrich Hustadt[2] and Renate A. Schmidt[1]

[1] Department of Computer Science, University of Manchester
Manchester M13 9PL, United Kingdom, {georgiel,schmidt}@cs.man.ac.uk
[2] Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom, U.Hustadt@csc.liv.ac.uk

Abstract. The guarded fragment and its extensions and subfragments have often been considered as a framework for investigating the properties of description logics. But there are other decidable fragments which all have in common that they generalise the standard translation of $\mathcal{ALC}$ to first-order logic. We provide a short survey of some of these fragments and motivate why they are interesting with respect to description logics.

In this abstract we focus on the description logic $\mathcal{ALC}$ and the extensions of $\mathcal{ALC}$ by disjunction, conjunction and negation on roles and discuss their relationship to less well known logics including Boolean modal logic, the two-variable fragment, the dual of the Maslov's class K, Quine's fluted logic, and the positive restrictive quantification fragment PRQ. We compare the expressive power of the formalisms, and discuss the applicability of various inference methods established in the field of non-classical logics to description logics.

We denote the extension of $\mathcal{ALC}$ by the role-forming operators $r_1, \ldots, r_n$ by $\mathcal{ALC}(r_1, \ldots, r_n)$. The definition of the standard semantics of $\mathcal{ALC}$ [8] and its extensions by the role operators disjunction, conjunction, negation and inverse, indicates that these languages can be considered as subfragments of first-order logic. One of these fragments is the guarded fragment [1]. However, one important property of the guarded fragment is that the guards, that is the atoms we obtain from the translation of roles, are always positive. Therefore, $\mathcal{ALC}(\neg)$ and its extensions contain concepts whose translation will not result in guarded formulae. An example of an $\mathcal{ALC}(\sqcap, \neg)$ concept which cannot be translated into a guarded formula is the set of cheese lovers, that is $\forall\neg(\mathit{likes} \sqcap \mathit{eats}).\neg \mathit{Cheese}$.

Making economical use of first-order variables concepts of $\mathcal{ALC}(\neg)$ and many other description logics can be embedded into finite-variable function-free fragments of first-order logic. One of them is the *two variable fragment* FO^2 which consists of those formulae of first-order logic that can be written using only two variables. The description logic which has the same expressive power as the two variable fragment is $\mathcal{ALCB}$ [4], i.e. $\mathcal{ALC}$ extended with full Boolean operations on roles and the inverse operator. An extension of FO^2 is the dual of Maslov's class K, denoted by $\overline{\mathrm{K}}$ [5]. Consider the following formula φ_1 which defines the concept *mwc* as a subset of married couples with a child: $\forall x \forall y(\mathit{mwc}(x,y) \rightarrow (\mathit{married}(x,y) \wedge \exists z(\mathit{has\_child}(x,z) \wedge \mathit{has\_child}(y,z))))$. The formula φ_1 is not in the guarded fragment and also not in FO^2, but it is in $\overline{\mathrm{K}}$. In contrast, the formula $\forall x \forall y(\mathit{mwd}(x,y) \rightarrow \forall z(\mathit{have\_child}(y,x,z) \rightarrow \mathit{doctor}(z)))$ which describes the concept *mwd* as a subset of married couples with a child who is a doctor, is guarded, but not in $\overline{\mathrm{K}}$. So, the guarded fragment is not a subfragment of $\overline{\mathrm{K}}$ nor is $\overline{\mathrm{K}}$ a subfragment of the guarded fragment. However, $\overline{\mathrm{K}}$ contains a variety of classical solvable fragments, namely the monadic class, the initially extended Skolem class, the Gödel class, and FO^2 as well as a range of non-classical logics, like a number of extended modal logics, many description logics, reducts of representable relational algebras [3].

The concept *mwmc* of married couples whose all children are married, defined as follows: $\forall x_1 \forall x_2(\mathit{mwmc}(x_1,x_2) \leftrightarrow (\mathit{married}(x_1,x_2) \wedge \forall x_3(\mathit{have\_child}(x_1,x_2,x_3) \rightarrow \exists x_4 \mathit{married}(x_3,x_4))))$ is not guarded but also not in $\overline{\mathrm{K}}$. It belongs to yet another solvable fragment of first-order logic, namely *fluted logic*. In fluted logic the relational atoms may be negated, and consequently $\mathcal{ALC}(\neg)$, $\mathcal{ALC}(\sqcup, \sqcap, \neg)$, and the Boolean modal logic which cannot be embedded into the guarded fragment can be embedded into fluted logic [7]. Moreover fluted logic can be extended with converse on relations while still preserving decidability [6]. Fluted logic with converse allows for the embedding of standard modal logics K, KT, KD, KB, KTB, more expressive logics, like $\mathcal{ALB}$ and the

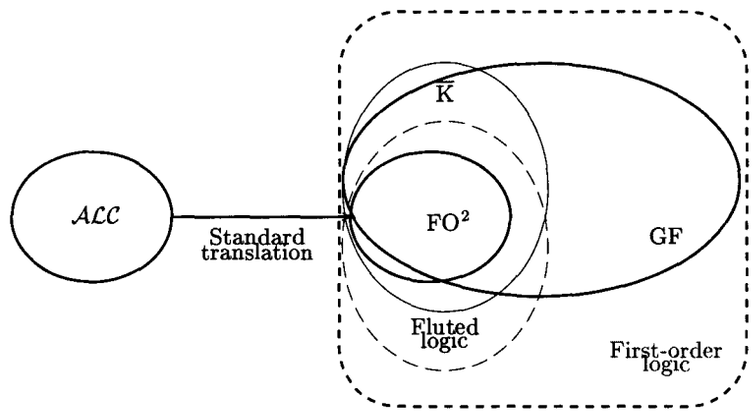

Fig. 1. The relationship of $\mathcal{ALC}$, FO^2, GF, $\overline{\mathrm{K}}$, and fluted logic

corresponding modal logic $K_{(m)}(\cap, \cup, \neg, ^\smile)$, and also FO^2. The relationship among the decidable fragments, discussed so far, in this abstract is summarised in Figure 1.

However, if instead of extensions of $\mathcal{ALC}$ by role-forming operators we look at products of $\mathcal{ALC}$ with modal logics, for example, basic modal logic, then there are examples which fall in neither of the solvable fragments of first-order logic we have looked at so far. Consider the sentence: *Minis are what Don believes to be a slow car*. The sentence is expressible in $K_{\mathcal{ALC}}$ as follows. $Minis \subseteq [Don]_{believes} \,.\, slow\_car$ Its standard translation to first-order logic is the formula φ_3.

$$\forall x(Minis(reality, x) \rightarrow \forall w(believes\_Don(reality, w) \rightarrow slow\_car(w, x)))$$

This formula is neither guarded nor fluted and is also not in $\overline{\mathrm{K}}$. However, it belongs to the so-called positive restrictive quantification fragment (PRQ) introduced in [2]. Even though PRQ is not solvable for fragments of PRQ that have the finite model property, there is a decision procedure in the form of an extended positive tableaux method [9]. The method does not only detect unsatisfiability but also generates finite models if they exist. Since $\mathcal{ALC}$ and many of its extensions have the finite model property, this procedure provides a general, sound, complete, and terminating method for solving the satisfiability problem for these logic without the necessity of additional correctness or termination proofs. The procedure is also applicable to products of modal and description logics which have the finite modal property such as $K_{\mathcal{ALC}}$ [10]. The product of $\mathcal{ALC}$ with basic modal logic K has the finite model property, therefore the considered example in $K_{\mathcal{ALC}}$ is covered by the procedure.

References

1. H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philos. Logic*, 27(3):217–274, 1998.
2. F. Bry and S. Torge. A deduction method complete for refutation and finite satisfiability. In *Proc. JELIA'98*, volume 1498 of *LNAI*, pages 1–17. Springer, 1998.
3. U. Hustadt and R. A. Schmidt. Maslov's class K revisited. In *Proc. CADE-16*, volume 1632 of *LNAI*, pages 172–186. Springer, 1999.
4. U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 191–205. Springer, 2000.
5. S. Ju. Maslov. The inverse method for establishing deducibility for logical calculi. In *Proc. Steklov Institute of Mathematics*, pages 25–96. American Mathematical Soc., 1971.
6. W. C. Purdy. Quine's 'limits of decision'. *J. Symbolic Logic*, 64(4):1439–1466, 1999.
7. R. A. Schmidt and U. Hustadt. A resolution decision procedure for fluted logic. In *Proc. CADE-17*, volume 1831 of *LNAI*, pages 433–448. Springer, 2000.
8. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *J. Artificial Intelligence*, 48:1–26, 1991.
9. S. Torge. *Überprüfung der Erfüllbarkeit im Endlichen: Ein Verfahren und seine Anwendung.* PhD thesis, Herbert Utz Verlag, München, 1998.
10. F. Wolter and M. Zakharyaschev. Multi-dimensional description logics. In *Proc. IJCAI'99*, pages 104–109. Morgan Kaufmann, 1999.

Using SPASS for proving theorems within Mereotopology

Shyamanta M Hazarika and **Anthony G Cohn**
School of Computing, University of Leeds
Leeds, United Kingdom.
{smh,agc}@comp.leeds.ac.uk

We report work in progress about using the automated theorem prover SPASS [12] for proving theorems within mereotopology. The mereotopological theory used is a spatio-temporal extension of the spatial representation language RCC-8 [11]. RCC-8 is a theory of space and time with a primitive dyadic relation of *connection*. The theory has been developed in a series of papers [11, 2, 6]. The most distinctive feature of the theory is that it uses extended regions instead of points as fundamental entities. For the formal theory to be used for reasoning about changes in spatial relations, it was augmented with a set of *envisioning axioms*. These axioms specify which direct transitions can be made in the topological relations between pairs of regions. The change of state in RCC relations via potential motion, analyzed through transition graphs in which relations form a *conceptual neighbourhood* was thus posited in [3] (see figure 2). Spatio-temporal continuity remained an implicitly assumed notion.

Galton [5] and thereafter Muller [10] looked at what continuity implies for a commonsense theory of motion. Muller had an explicit generic characterization of continuity and presented theorems for the (non-existence of) transitions between states in RCC-8 relations [10] within a space-time framework. But Muller's interpretation of theorems of (non-existence of) transitions has been shown not to be fully adequate [4]. Davis presents an alternative framework for characterisation of transitions. However it is not expressed as an object level first-order theory, and sacrifices the spirit of mereotopology as it defines time instants and spatio-temporal points.

We present a more comprehensive framework. The basic entities of our theory are non empty regions of space-time. Following [7], space-time regions are termed *histories*. Note that following previous work within the group [11, 6] we do not wish to admit lower dimensional entities - e.g. in our work on spatial mereotopology all regions were of the same dimension and we did not consider boundaries as spatial entities. Here too we do not admit lower dimensional entities such as temporal points into our ontology for the same reasons as argued in [11, 2]. In order to refer to regions within a given time we introduce the notion of *temporal slice*. Thus $\mathsf{TS}(x, y)$ says that x is a temporal slice of y and the syntactic sugar $\frac{y}{t}$ denotes the temporal slice of y during t. Fig. 1 shows a space-time history for a 2-D object and a temporal slice of the history. In a n-D space, a space-time history is a n+1 dimensional volume.

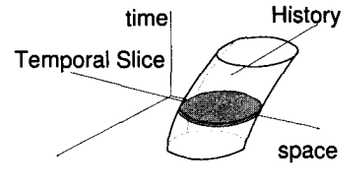

Figure 1: A s-t history is a n+1 dimensional volume in a n-D space.

Our theory is based on the basic dyadic relation of connection. We have three versions of connection relation: spatial, temporal and spatio-temporal interpreted in pure space, time and space-time respectively (see [8] for details). We introduce operators to characterize transitions. Furthere we have characterized a number of distinct notions of space-time continuity [8] including the notion of *strong firm-continuity* on which figure 2 depends.

The extended spatio-temporal theory is a first-order theory with equality. To formalize our theory we use a sorted first-order logic based on the logic LLAMA [1]. The principal sorts we use are REGION and NULL. We define a set of Boolean functions. The Boolean functions are sum, difference, intersection and complement. The functions are partial but as discussed in [2], are made total in the sorted logic by specifying sortal restrictions and letting the result sort of the partial functions be REGION $\cup$ NULL. The functions can be regarded as genuine Boolean operators

over the domain REGION ∪ NULL. In the SPASS input files along with the definitions for the Boolean functions (mentioned here), we add axioms for an equational theory of Boolean algebra [adapted from [9]] and have axioms to link the Boolean algebra to the relational part of the theory. This allows axioms for the Boolean functions to be added without loss of saturation. Although previously [8] we defined $\frac{y}{t}$ as syntactic sugar, for computational reasons, in SPASS $\frac{y}{t}$ is represented using an actual binary function, with axioms to specify its semantics.

In the absence of intended models and a syntactic proof of completeness, the only way to know that the axiomatisation fulfills our intentions to characterize continuous transitions is by proving 'transition theorems'. There are two aspects to proving the correctness of the conceptual neighbourhood: the links that are present need to be shown to be necessary, and those absent to be shown to represent discontinuous transitions. The latter is a theorem proving task but the former requires model building and appears to be much harder automated. We confine ourselves to the second task here.

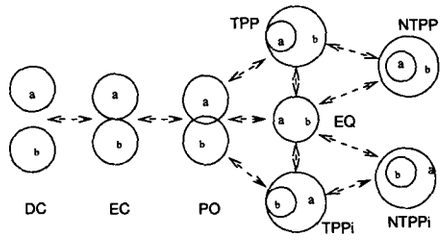

Figure 2: Conceptual Neighbourhood for RCC-8 relations under strong firm continuity.

The *non-existence theorems* formally show which transitions are not continuous and correspond to the links absent in the transition graph shown in figure 2 for RCC-8 under strong frim continuity. We are using the automated theorem prover SPASS to prove the non-existence transition theorems. The clausal form of the underlying theory consists of 571 clauses, 80 predicates and 219 function symbols. http://www.comp.leeds.ac.uk/qsr/SpassProofs list some proofs found by SPASS in our spatio-temporal theory.

Acknowledgements

The financial assistance of the EPSRC under grant GR/M56807 is gratefully acknowledged. The first author acknowledge the CWSC for financial assistance under reference INCS-1999-177.

References

[1] A G Cohn, 'A more expressive formulation of many sorted logic', *Journal of Automated Reasoning*, **3**, 113–200, (1987).

[2] A G Cohn, B Bennett, J Gooday, and N Gotts, 'RCC: A Calculus for Region based Qualitative Spatial Reasoning', *GeoInformatica*, **1**(3), 275–316, (1997).

[3] A G Cohn, N M Gotts, Z Cui, D A Randell, B Bennett, and J M Gooday, 'Exploiting temporal continuity in qualitative spatial calculi', in *Spatial and Temporal Reasoning in Geographical Information Systems*, ed., M J Egenhofer et al., pp. 5–24. Oxford University Press, (1998).

[4] E Davis, 'Continuous shape transformation and metrics of shape', *Fundamenta Informaticae*, **46**(1-2), 31–54, (2001).

[5] A P Galton, *Qualitative Spatial Change*, Oxford University Press, 2000.

[6] N M Gotts, J M Gooday, and A G Cohn, 'A connection based approach to common-sense topological description and reasoning', *The Monist*, **79**(1), 51–75, (1996).

[7] P J Hayes, 'The Second Naive Physics Manifesto', in *Formal Theories of the Commonsense World*, ed., J. R. Hobbs et al., 1–36, Ablex, (1985).

[8] S M Hazarika and A G Cohn, 'Qualitative spatio-temporal continuity', in *Proc. of COSIT'01*, number 2205 in LNCS, pp. 92–107, (2001).

[9] K. Kuratowski, *Introduction to Set Theory and Topology*, Pergamon Press, 2nd edn., 1972.

[10] P. Muller, 'A qualitative theory of motion based on spatio-temporal primitives', in *Proc. of KR'98*, ed., A. G. Cohn et al., pp. 131–141, (1998).

[11] D. A. Randell, Z. Cui, and A. G. Cohn, 'A spatial logic based on regions and connection', in *Proc. of KR'92*, pp. 165–176, (1992).

[12] C. Weidanbach, 'SPASS: Combining superposition, sorts and splitting', in *Handbook of Automated Reasoning*, eds., A Robinson and A Voronkov, Elsevier Sciene Publishers. SPASS is available from `http://spass.mpi-sb.mpg.de/`.

Fast Normalization in the HOL Theorem Prover

Joe Hurd*
Computer Laboratory
University of Cambridge
joe.hurd@cl.cam.ac.uk

1 March 2002

1 Introduction

The 'LCF design' inherited by the HOL theorem prover [1] allows users to write ML functions encoding arbitrary patterns of logical deduction. Such ML functions are called derived rules,[1] because the only way that they can create theorems is by composing a sequence of existing rules. The initial rules, implemented in a small logical kernel, are the primitive inference rules of higher-order logic. This design philosophy is called fully-expansive proof, because every derived rule can be 'fully expanded' to a (long) sequence of primitive inferences.

Example: Harrison's HOL implementation of the model elimination procedure is a derived rule. It takes a goal term, performs a proof search, and then executes the successful proof as a sequence of primitive inferences. □

We are interested in creating efficient derived rules for normalizing terms. Converting terms to conjunctive normal form (CNF) is necessary to apply many first-order proof procedures, and some decision procedures for Presburger arithmetic transform terms to disjunctive normal form as part of their operation. Our present focus is on converting terms to definitional CNF: a normal form similar to CNF in which the result formulas are only linearly larger than the input formulas. An upper bound on the result size is essential when normalizing large propositional formulas (such as those generated by hardware verification) for input to satisfiability (SAT) solvers. In our experiments, we use Gordon's `HolSatLib`[2] library for hol98. It provides a harness for invoking SAT solvers on HOL terms, and currently supports `SATO`, `GRASP` and `zCHAFF`.

Example: To prove that the boolean 'exclusive or' operation is associative, we

*Supported by EPSRC project GR/R27105/01.
[1] Or tactics, depending on the context in which they are used.
[2] `http://www.cl.cam.ac.uk/users/mjcg/HolSatLib/`

first convert the negation of the goal to definitional CNF:

$$\begin{aligned}\vdash\ & \neg((\neg(p = \neg(q = r)) = \neg(\neg(p = q) = r))) = \\ & \exists v_0, v_1, v_2, v_3, v_4. \\ & \quad (v_4 \vee v_1 \vee v_3) \wedge (v_4 \vee \neg v_1 \vee \neg v_3) \wedge (v_1 \vee \neg v_3 \vee \neg v_4) \wedge (v_3 \vee \neg v_1 \vee \neg v_4) \wedge \\ & \quad (v_3 \vee v_2 \vee r) \wedge (v_3 \vee \neg v_2 \vee \neg r) \wedge (v_2 \vee \neg r \vee \neg v_3) \wedge (r \vee \neg v_2 \vee \neg v_3) \wedge \\ & \quad (v_2 \vee p \vee \neg q) \wedge (v_2 \vee \neg p \vee q) \wedge (p \vee q \vee \neg v_2) \wedge (\neg q \vee \neg p \vee \neg v_2) \wedge \\ & \quad (v_1 \vee p \vee v_0) \wedge (v_1 \vee \neg p \vee \neg v_0) \wedge (p \vee \neg v_0 \vee \neg v_1) \wedge (v_0 \vee \neg p \vee \neg v_1) \wedge \\ & \quad (v_0 \vee q \vee r) \wedge (v_0 \vee \neg q \vee \neg r) \wedge (q \vee \neg r \vee \neg v_0) \wedge (r \vee \neg q \vee \neg v_0) \wedge v_4\end{aligned}$$

Next we strip off the existential quantifiers (the 'definitions' of definitional CNF) from the RHS of this equation, and invoke `HolSatLib` on the resulting term. The HOL term is then converted to DIMACS format, sent to a SAT solver, and, if no satisfiable assignments are found, the negation of the term is asserted as a HOL theorem (with a 'SAT solver' oracle tag[3]). Finally, a couple more primitive inferences suffice to derive our original goal as a theorem. □

As can be observed, the above method of proving propositional formulas involves a conversion to definitional CNF. It is relatively easy to generate the desired term, and so we can simply create the conversion theorem with a 'normalization' oracle tag. However, it is interesting to see how efficiently we can construct the fully-expansive proof of the conversion theorem. We could naively try to mirror the definitional CNF algorithm using primitive inferences, but large intermediate terms make this algorithm is quadratic in the size of the input term. Instead, we use a technique of Harrison [2] using variable vectors to create a novel algorithm that runs in (nearly) linear time. On our `ADD4` example term,[4] this uses more primitive inferences than the naive method, but small terms at each intermediate point mean that the overall time taken is lower.

Operation on the `ADD4` Term	**Time (s)**	**Infs.**
Definitional NNF	10.610	7677
Oracle version of definitional CNF	0.390	0
Naive definitional CNF	122.620	12034
Definitional CNF using variable vectors	28.800	238258
Applying the `zCHAFF` solver	4.680	0

References

[1] M. J. C. Gordon and T. F. Melham. *Introduction to HOL (A theorem-proving environment for higher order logic).* Cambridge University Press, 1993.

[2] John Harrison. Binary decision diagrams as a HOL derived rule. *The Computer Journal*, 38:162–170, 1995.

[3] Theorem tags are propagated by the primitive inferences, and so for each theorem it is easy to discover the tools that were used as oracles in the proof.

[4] The `ADD4` term arose from hardware verification, and contains 1111 atoms.

An Investigation into Proof Automation for the SPARK Approach to High Integrity Ada

Andrew Ireland Bill J. Ellis Julian Richardson

Department of Computing & Electrical Engineering
Heriot-Watt University
Edinburgh, Scotland
E-mail: a.ireland@hw.ac.uk

Position Statement

We are currently investigating[1] the application of proof planning to the SPARK[2] approach to developing high integrity Ada. The SPARK approach advocates *correctness by construction* – where the development of code and verification proofs go hand-in-hand. Proof planning involves the use of high-level patterns of reasoning in constraining the automatic search for verification proofs.

Our investigation began in September 2001. An evaluation of the potential use of existing proof plans within this new application is currently under-way. This evaluation will also inform our development of new proof plans. In addition, we are currently building a first prototype of NuSPADE, an experimental proof planning system. NuSPADE will extend the SPADE Proof Checker[3], an interactive theorem proving environment designed to support the proof of verification conditions arising from software written in SPARK. This extension will build upon earlier work implemented within the Clam proof planning system. We believe that NuSPADE will significantly increase the level of proof automation that is currently supported by the SPADE Proof Checker.

[1]The research outlined here is funded under the EPSRC Critical Systems Programme – EPSRC grant GR/R24081.

[2]Praxis Critical Systems Ltd – `http://www.praxis-cs.co.uk`

[3]The SPADE Proof Checker is part of the SPARK Examiner tool-set and is a product of Praxis Critical Systems Ltd.

Automatic Learning of Proof Methods in Proof Planning*

Mateja Jamnik[1] Manfred Kerber[1]
Martin Pollet[1,2] Christoph Benzmüller[2]

[1]School of Computer Science, The University of Birmingham
Birmingham B15 2TT, England, UK
www.cs.bham.ac.uk/~{mxj|mmk|mxp}

[2]Fachbereich Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany, www.ags.uni-sb.de/~{pollet|chris}

Abstract

We present a framework for automated learning within mathematical reasoning systems. In particular, this framework enables proof planning systems to automatically learn new proof methods from well chosen examples of proofs which use a similar reasoning pattern to prove related theorems. Our framework consists of a representation formalism for methods and a machine learning technique which can learn methods using this representation formalism. We briefly present how to learn methods in the ΩMEGA proof planner. The approach is a two-stage one, first to learn method outlines and second to build complete methods from method outlines.

Proof planning [Bun88] is an approach to theorem proving which uses proof methods rather than low level logical inference rules to find a proof of a theorem at hand. A proof method specifies a general reasoning pattern that can be used in a proof, and typically represents a number of individual inference rules. Proof planners search for a proof plan of a theorem which consists of applications of several methods. One of the ways to extend the power of a proof planning system is to enlarge the set of available proof methods. This is particularly beneficial when a class of theorems can be proved in a similar way, hence a new proof method can encapsulate the general reasoning pattern of a proof for such theorems. A difficulty in applying

*This work was supported by EPSRC grant GR/M22031 and European Commission IHP Calculemus Project grant RTN1-1999-00301.

a proof pattern to many domains is that in the current proof planning systems new methods have to be implemented and added by the developer of a system. We devised a framework within which a proof planning system can learn frequently occurring patterns of reasoning automatically from a number of typical examples, and then use them in proving new theorems [JKB01]. The availability of such patterns, captured as proof methods in a proof planning system, reduces search and proof length. We implemented this learning framework for the proof planner ΩMEGA [BCF+97], and – we call our system LEARNΩMATIC. LEARNΩMATIC consists of, first, a learning mechanism, which learns so-called *method outlines* from proof traces of typical examples. Method outlines are essentially names of methods, sequences, disjunctions, repetitions (arbitrary or fixed number), and branching points (as list of branches). For instance, from the outlines $e_1 = [a, a, a, a, b, c]$ and $e_2 = [a, a, a, b, c]$ the system would generalise to $[[a]^*, [b, c]]$.

The second part of LEARNΩMATIC consists of a mechanism which enables ΩMEGA to use these method outlines as fully fleshed proof methods by adding to them additional information and performing search for information that cannot be reconstructed.

Further information about the project and links to publications can be found on `www.cs.bham.ac.uk/~mmk/projects/MethodFormation/`.

References

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In W. McCune, editor, *Proceedings of the 14th International Conference on Automated Deduction (CADE)*, pages 252–255, Townsville, Austalia, 1997. Springer Verlag. LNAI 1249.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing Lusk and Ross Overbeek, editors, *Proc. of the 9th CADE*, pages 111–120, Argonne, Illinois, USA, 1988. Springer Verlag, LNCS 310.

[JKB01] Mateja Jamnik, Manfred Kerber, and Christoph Benzmüller. Learning method outlines in proof planning. Technical Report CSRP-01-08, School of Computer Science, The University of Birmingham, Birmingham, England, UK, 2001.

TRP++: Implementation of Clausal Resolution for Propositional Linear-Time Temporal Logic

Boris Konev*

Logic and Computation Group, Department of Computer Science
University of Liverpool, Liverpool L69 7ZF, UK
B.Konev@csc.liv.ac.uk

1 Introduction

Temporal logics are extensions of classical logic, with operators that deal with time. They have been used in a wide variety of areas within Computer Science and Artificial Intelligence, for example robotics [18], databases [19], hardware verification [11] and agent based systems [17]. In particular, propositional temporal logics have already made significant impact within Computer Science, having been applied to:

- the specification and verification of reactive (e.g. distributed or concurrent) systems [15];
- the synthesis of programs from temporal specifications [16, 14];
- the semantics of executable temporal logic [8, 9];
- algorithmic verification via model-checking [10, 2]; and
- knowledge representation and reasoning [4, 1, 20].

In developing these techniques, temporal proof is often required, and we base our work on practical proof techniques on the clausal resolution approach to temporal logic.

The clausal resolution method for propositional linear-time temporal logics provides the basis for a number of temporal provers. The method is based on an intuitive clausal form, called SNF, comprising 3 main clause types and a small number of resolution rules [5]. While the approach has been shown to be competitive [12, 13], we aim at an even more efficient implementation.

As usual, the temporal resolution approach aims at deduction of a contradiction from a given set of formulas.

2 Separated Normal Form

A *propositional temporal problem*, whose satisfiability we are interested in, consists of four parts[1]:

1. a universal part $\mathcal{U}$, given by a set of propositional clauses;
2. an initial part $\mathcal{I}$ with the same form as the universal part;
3. a step part $\mathcal{S}$, given by a set of propositional step temporal clauses of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_k \Rightarrow \bigcirc(M_1 \vee M_2 \vee \ldots \vee M_l) \qquad (\textit{step} \text{ clause}),$$

   where L_i, M_j are literals;
4. an eventuality part $\mathcal{E}$, given by a set of propositional eventuality temporal clauses of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_k \Rightarrow \Diamond(M),$$

   where L_i, M are literals.

We say that a problem is satisfiable if, and only if, the formula $\mathcal{I} \wedge \Box\mathcal{U} \wedge \Box\mathcal{S} \wedge \Box\mathcal{E}$ is satisfiable.

It is known [5] that a PLTL-formula is satisfiable if, and only if, a set of temporal clauses is satisfiable. To translate a temporal formula into the SNF form (see [5] for full details), we essentially apply a set of the transformation rules based upon the renaming of complex expressions by new propositions and upon the substitution of temporal operators by their fixpoint definitions.

* On leave from Steklov Institute of Mathematics at St.Petersburg

[1] The **TRP++** system uses slightly different normal form than described in [5] in that we allow universal clauses; each universal clause C can be represented by a pair $\{\mathbf{start} \Rightarrow C, \mathbf{true} \Rightarrow \bigcirc(C)\}$ in terms of [5].

3 Temporal resolution

The deduction system for temporal problems essentially consist of the step resolution rule (that could be thought of as an ordinary resolution rule acting on "arithmetization" of the initial, universal, and step parts) and the eventuality resolution rule that involves the detection of a set of formulas that together imply $\Box\neg l$ (known as a *loop in* $\neg l$) for resolution with $\Diamond l$. Note that for arithmetization, we represent temporal clauses as (a very restrictive form of) first order clauses, namely, an initial clause C will be represented as $C(0)$, a universal clause D as $D(x)$, and a step clause $(\bigwedge L_i \Rightarrow \bigcirc(\bigvee M_j))$ as $(\bigvee \neg(L_i(x)) \vee \bigvee(M_j(f(x)))$.

Then, any first-order resolution system would allow us to perform step resolution steps; however, due to very restrictive nature of the formulas, **TRP++** uses it's own "near propositional" approach to deal with them. Further, search for loops (that is, the most costly part of the method) can be implemented by similar techniques to those described in [3].

4 Acknowledgment

The author would like to thank A. Degtyarev, C. Dixon, U. Hustadt, and M. Fisher for fruitful discussions about the subject. The author would like to thank the both EPSRC (via grant GR/L87491) and the University of Liverpool (via Research Development Funding) for their support in this work.

References

1. A. Artale and E. Franconi. Temporal description logics. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence.* To appear.
2. E. Clarke, O. Grumberg and D. A. Peled. *Model Checking.* MIT Press, 2000.
3. C. Dixon Temporal Resolution using a Breadth-First Search Algorithm. in *Annals of Mathematics and Artificial Intelligence.* volume 22, pages 87–115, special issue on Temporal Representation and Reasoning 1998. Baltzer Science Publishers. ISSN 1012-2443.
4. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge.* MIT Press, 1996.
5. M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computation Logic,* 2(1), January 2001.
6. M. Fisher. A resolution method for temporal logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91).* Morgan Kaufman, 1991.
7. M. Fisher. A normal form for first-order temporal formulae. In Proc.of 11th International Conference on Automated Deduction(CADE'11), volume 607 of *Lecture Notes in Computer Science.* Springer Verlag, 1992.
8. M. Fisher. An introduction to executable temporal logics. *Knowledge Engineering Review,* 11:43–56, 1996.
9. M. Fisher. A temporal semantics for concurrent METATEM. *J. Symbolic Computation,* 22(5/6):627–648, 1996.
10. G.J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, New Jersey, 1991.
11. G. J. Holzmann. The Model Checker Spin. *IEEE Trans. on Software Engineering,* 23(5):279–295, May 1997. Special Issue on Formal Methods in Software Practice.
12. U. Hustadt and R. Schmidt. Scientific Benchmarking with Temporal Logic Decision Procedures *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002),* 2002 (to appear).
13. U. Hustadt and R. Schmidt. Formulae which Highlight Differences between Temporal Logic and Dynamic Logic Provers In *Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics.* pp.68–76, Siena, Italy, 2001.
14. O. Kupferman and M. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic.* pp.109-128. Kluwer, 2000.
15. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer, 1992.
16. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. Princip. Prog. Lang.,* pp.179–190, 1989.
17. A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation,* 8(3):293–342, 1998.
18. M. P. Shanahan. *Solving the Frame Problem.* MIT Press, 1997.
19. A. Tansel, editor. *Temporal Databases: theory, design, and implementation.* Benjamin/Cummings, 1993.
20. F. Wolter and M. Zakharyaschev. Temporalizing description logics. In *Frontiers of Combining Systems II,* pp.379–401. Research Studies Press LTD, England, 2000.

Temporalising tableaux

Roman Kontchakov,[1] Carsten Lutz,[2] Frank Wolter,[3] and Michael Zakharyaschev[1]
[1]Department of Computer Science, King's College,
Strand, London WC2R 2LS, U.K.
[2]LuFG Theoretical Computer Science RWTH Aachen,
Ahornstr. 55, 52074 Aachen, Germany.
[3]Institut für Informatik, Universität Leipzig
Augustus-Platz 10-11, 04109 Leipzig, Germany.
(e-mails: {romanvk,mz}@dcs.kcl.ac.uk, lutz@cs.rwth-aachen.de, wolter@informatik.uni-leipzig.de)

First-order temporal logic (FOTL) based on the flow of time $\langle \mathbb{N}, < \rangle$ is notorious for its 'bad computational behaviour:' even the two-variable monadic fragment of this logic is not recursively enumerable (see e.g. [2] and references therein). A certain breakthrough has been recently achieved in [2], where the so-called *monodic fragment* of FOTL is introduced by restricting applications of temporal operators to formulas with at most one free variable. The full monodic fragment (containing full first-order logic) turns out to be axiomatisable [3]. Moreover, by restricting its first-order part to decidable fragments, we obtain decidable monodic FOTLs, say, the monodic guarded, monodic two-variable, and monodic monadic fragments. This opens a way to various applications of the monodic FOTL in knowledge representation, temporal databases, and other fields. For example, many temporal description logics and spatio-temporal logics can be regarded as fragments of monodic FOTL [4, 5, 6]. Unfortunately, the decision procedures provided in [2] are of model-theoretic character and cannot be used as a basis for implementations. In [1] a resolution-based approach was developed for fragments weaker than monodic fragments. A *tableau-based analysis* of the decision problem for monodic FOTL has been missing. In this paper we are trying to fill in this gap. More specifically, our aims are as follows:

1. *to develop a general framework for devising tableau-based decision procedures for decidable monodic FOTLs and then,*
2. *within this framework, to construct tableau systems for a number of concrete monodic fragments.*

We consider monodic FOTLs interpreted in models with both expanding and constant domains. The former case is technically much easier, but the latter one is more general: reasoning with expanding domains can be reduced to reasoning with constant domains.

Our approach is based on the following ideas:

- *modularity*—a decision procedure for a given fragment of first-order logic is combined with Wolper's tableaux [7] for propositional temporal logic (PTL);
- *finite quasimodel* representations of temporal models with potentially infinite first-order domains;
- the *minimal type* technique to keep the domains of temporal models constant.

To achieve modularity, we separate the temporal and the pure first-order parts of formulas and treat them using available procedures for PTL and the corresponding first-order fragment. This is done

by replacing all occurrences of formulas starting with temporal operators by their 'surrogates'—unary predicates (the language is monodic) the proper 'temporal behaviour' of which is ensured by some auxiliary *surrogate axioms*. The resulting set of purely first-order formulas is treated then by the given first-order decision procedure to obtain descriptions of all possible models for this set. If the procedure fails, then the formula is not satisfiable. Otherwise we make one 'temporal step,' namely, omit the 'next-time' operator (as in Wolper's tableaux) and add new surrogate axioms. This yields another set of purely first-order formulas, an so forth.

Some special techniques are used to preserve the representation of tableaux finite and to guarantee termination. For example, first-order models are represented by finite sets of *types*, each of which standing for possibly infinite number of domain elements. *Quasimodels* are used to encode temporal models by associating a finite set of types with each time instant.

When a tableau is completed, the pruning technique (again, as in Wolper's tableaux) is used to ensure that all eventualities are fulfilled. Then the resulting tableau represents quasimodels of the testing formula.

A rather general theorem provides conditions under which a first-order decision procedure can be combined with Wolper's tableaux to yield a tableau-based decision procedure for the corresponding monodic FOTL. The price we have to pay for this level of generality is that the resulting combined tableaux are far from optimal. In particular, in many concrete cases new rules can be used instead of surrogate axioms. Thus, the general framework for combining tableaux is not supposed for direct applications or implementations, but rather as a *guide* for considering more specific cases.

Acknowledgements

The work of the first and fourth author was partially supported by UK EPSRC grant GR/R45369/01 "Analysis and mechanisation of decidable first-order temporal logics." The work of the third author was partially supported by Deutsche Forschungsgemeinschaft (DFG) grant Wo583/3-1.

References

[1] A. Degtyarev and M. Fisher. Towards First-order Temporal Resolution. *KI2001: Advances in Artificial Intelligence*, F. Baader, G. Brewka and T. Eiter, editors, LNAI 2174, Springer, pp.18–32, 2001

[2] I. Hodkinson, F. Wolter and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Apllied Logic*, 106:85–134, 2000.

[3] F. Wolter and M. Zakharyaschev. Axiomatizing the monodic fragment of first-order temporal logic. *Annals of Pure and Apllied Logic*, 2002. (In print.)

[4] I. Hodkinson, F. Wolter and M. Zakharyaschev. Monodic fragments of first-order temporal logics: 2000–2001 A.D. In *Logic for Programming, Artificial Intelligence and Reasoning*, number 2250 of LNAI, Springer 2001, pp.1–23.

[5] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2002. (To appear.)

[6] F. Wolter and M. Zakharyaschev. Qualitative spatio-temporal representation and reasoning: a computational perspective. "Exploring Artifitial Intelligence in the New Millenium," Morgan Kaufmann, 2002.

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

The decidability of the first-order theory of the Knuth-Bendix orders in the case of unary signatures

Konstantin Korovin and Andrei Voronkov

University of Manchester
{korovin,voronkov}@cs.man.ac.uk

Introduction of ordering constraints has been one of the main breakthroughs in the saturation based theorem proving. Using solvability of ordering constraints we can dramatically reduce the number of redundant inferences in a resolution-based prover. As a consequence, the problem of solving ordering constraints for the known simplification orders is one of the important problems in the area. A simplification order is a total monotonic order on ground terms. Given such an order we can consider ordering constraints which are quantifier-free formulas in the language of the term algebra with equality and order. Two kinds of orders are used in automated deduction: the Knuth-Bendix orders [5] and various versions of the recursive path orders [3]. Because of its importance, the decision problem for ordering constraints has been well-studied. For the recursive path orders decidability and complexity issues were considered in [4, 1, 11, 12, 10, 9]. For the Knuth-Bendix orders we have the following results: the decidability of constraints [6], a nondeterministic polynomial-time algorithm for constraint solving [7], a polynomial-time algorithm for solving constraints consisting of a single inequality [8].

In resolution-based theorem proving there are important simplifications which allow us to remove clauses form the search space (for example subsumption). It turns out that in order to express applicability conditions for these simplifications, we need to consider constraints which involve first-order quantifiers. Unfortunately the first-order theory of the recursive path orders is undecidable [13, 2]. Only recently the decidability of the first-order theory of recursive path orders in the case of unary signatures has been proven [9]. A signature is called unary if it consists of unary function symbols and constants.

Continuing our work on the Knuth-Bendix orders we prove the decidability of the first-order theory of the Knuth-Bendix orders in the case of unary signatures.

Theorem 1. *The first order theory of any Knuth-Bendix order for a unary signature is decidable.*

Our decision procedure uses interpretation of unary terms as trees and uses decidability of the weak monadic second-order theory of binary trees.

Let us note that the decidability problem for the first-order theory of the Knuth-Bendix orders in the the case of arbitrary signatures remains open.

References

1. H. Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.
2. H. Comon and R. Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theoretical Computer Science*, 176(1-2):67–87, 1997.
3. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
4. J.-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. *Lecture Notes in Computer Science*, 510:455–468, 1991.
5. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.
6. K. Korovin and A. Voronkov. A decision procedure for the existential theory of term algebras with the Knuth-Bendix ordering. In *Proc. 15th Annual IEEE Symp. on Logic in Computer Science*, pages 291–302, Santa Barbara, California, June 2000.
7. K. Korovin and A. Voronkov. Knuth-bendix ordering constraint solving is NP-complete. In F. Orejas, P.G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001*, volume 2076 of *Lecture Notes in Computer Science*, pages 979–992. Springer Verlag, 2001.
8. K. Korovin and A. Voronkov. Verifying orientability of rewrite rules using the Knuth-Bendix order. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA 2001*, volume 2051 of *Lecture Notes in Computer Science*, pages 137–153. Springer Verlag, 2001.
9. Narendran and Rusinowitch. The theory of total unary rpo is decidable. In *First International Conference on Computational Logic*, 2000.
10. Narendran, Rusinowitch, and Verma. RPO constraint solving is in NP. In *CSL: 12th Workshop on Computer Science Logic*. LNCS, Springer-Verlag, 1998.
11. R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.
12. R. Nieuwenhuis and J.M. Rivero. Solved forms for path ordering constraints. In *In Proc. 10th International Conference on Rewriting Techniques and Applications (RTA)*, volume 1631 of *Lecture Notes in Computer Science*, pages 1–15, Trento, Italy, 1999.
13. Treinen. A new method for undecidability proofs of first order theories. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 10, 1990.

CONCEPTOOL: Intelligent Support to Knowledge Management

Helmut Meisel Ernesto Compatangelo
Department of Computing Science, University of Aberdeen
{hmeisel, compatan}@csd.abdn.ac.uk

Abstract

We are developing an Intelligent Knowledge Management Environment called CONCEPTOOL, which has been explicitly conceived to support knowledge analysis and sharing at the conceptual level using both logic-based and heuristic deductions. At the moment of writing, CONCEPTOOL provides core analysis functionalities for an expressive enhanced entity relationship model. Heuristic deductions beside purely logic-based ones, as well as the possibility of selectively de-activating specific components of the expressive power, are among the novel features of our approach.

1 Motivations and rationale

Knowledge Management (KM) develops formal frameworks for knowledge construction, access and reuse. KM approaches are based on the concept of a *knowledge lifecycle*, which encompasses knowledge acquisition, modelling, retrieval, publishing, maintenance and reuse. These activities are considered as the explicit research challenges of the Advanced Knowledge Technologies (AKT) Consortium, which aims at producing an integrated approach to the knowledge lifecycle that encompasses a multi-disciplinary range of viewpoints (AKT Consortium, 2001).

Effective support for the knowledge lifecycle requires analysis and sharing functionalities based on a combination of logical and heuristic inferences. Analysis includes knowledge verification, validation and augmentation services. Sharing includes knowledge translation and alignment services that enable the same interpretation of (parts of) a Knowledge Base (KB) by different humans or computer agents (Compatangelo and Sleeman, 2001).

Despite the sizeable number of available systems for knowledge analysis and sharing, there is still a lack of integrated environments which provide a comprehensive set of flexible inferential services for a variety of heterogeneous knowledge models. In fact, several systems only provide knowledge editing and presentation (i.e. structuring) functionalities for semi-formal conceptual models. Some systems provide a core set of knowledge analysis functionalities for carefully selected fragments of First-Order Logic (FOL). A few systems provide knowledge sharing functionalities for expressive subsets of FOL.

Structuring tools generally lack analysis and sharing functionalities, which are implicitly based on a formal knowledge model. Analysis tools mainly focus on epistemological knowledge (e.g. frames), lacking support for conceptual knowledge (e.g. entities, functions) and its sharing. Sharing tools generally deal with the interchange of logical or epistemological knowledge, lacking support for the analysis of conceptual knowledge.

In order to overcome these limitations, we have been developing an Intelligent Knowledge Management Environment (IKME) called CONCEPTOOL, which provides specialised analysis and sharing functionalities directly at the conceptual level. Our approach uses a combination of underlying terminological and heuristic reasoning services to support lifecycle activities for declarative KBs (e.g. conceptual schemas or ontologies). In doing so, we adapt existing modelling and reasoning techniques to the needs of sharing-driven analysis, without being conditioned by their oddities or limited by their drawbacks.

2 System overview

We have planned the development of CONCEPTOOL in stages, progressively increasing the number of specialised services and the expressive power of its formal knowledge model. At the time of writing, with stage 1 fairly completed, CONCEPTOOL provides structuring and analysis functionalities for reasoning with an expressive Enhanced Entity-Relationship (EER) model. The EER language presently recognised by CONCEPTOOL includes:

- Entity identifiers (simple and composite ones, as well as external identifiers for weak entities).
- Entity and relationship attributes, with the possibility of specifying standard (primitive) attribute domains, fillers and full multiplicity ranges.
- Multiple *is_a* links (i.e. multiple hierarchical generalisation-specialisation links) between entities as well as between relationships of the same arity.
- Full range of cardinality constraints for the participation of an entity in a relationship, without being limited to the four cases 0:1, 1:1, 0:N, 1:N.
- Separate disjointness and full coverage assertions among entities.

Selectable identifiers, attribute fillers, full participatory ranges and a conceptually unambiguous definition of multiple hierarchical links between relationships enable knowledge modelling and analysis within the EER framework. Therefore, the expressive power of CONCEPTOOL represents an improvement over that of other comparable systems for EER conceptual modelling and analysis, such as iCOM (Franconi and Ng, 2000).

Our approach to knowledge analysis in CONCEPTOOL is that of providing a whole range of reasoning services at the conceptual level. Therefore, CONCEPTOOL has been designed to provide both logic-based and heuristic deductions. Our EER conceptual language has been mapped (with suitable extensions and emulations) into the epistemological language $\mathcal{SHIQ}$ recognised by the Description Logic (DL) reasoner iFaCT (Horrocks, 1999).

Logic analysis does not make use of sources of knowledge (such as ontologies, lexicons, heuristic rules) which are external to the KB under investigation. Therefore, it could fail to detect relationships between semantically similar concepts in a KB if these concepts are described using different terminologies. In fact, the lack of subsumption between concepts with pairwise different (but semantically equivalent) role names is a well-known major limitation of terminological approaches.

The above limitation can be overcome by lexical and heuristic analysis, which provide suggestions likely to be meaningful (but not always necessarily true) in the analysed context. As part of a more sophisticated service still under development, CONCEPTOOL presently provides a simple customisable heuristic analysis mechanism based on substring matching between two or more entity names.

In addition to syntactic analysis, CONCEPTOOL offers the following logic-based semantic analysis services:

- EER schema consistency.
- Detection of incoherent entities and relationships.
- Detection of synonyms in entities or relationships.
- Propagation of properties and constraints.
- Derivation of parents and ancestors.
- Derivation of consequences of assertions.

In order to improve the detection of new links between concepts, CONCEPTOOL supports a full customisation of its expressive power, with the possibility of selectively de-activating or re-interpreting specific constructors. More specifically, the distinction between attributes and identifiers can be de-activated, so that the latter are re-interpreted as attributes. The domain of identifiers and attributes, as well as the multiplicity of attributes, can be redefined, so that all of them share the same values. Either just attribute fillers or all attribute properties can be ignored at once, such that attribute names only are taken into account. Moreover, assertions (disjointness and coverage) and relationships can be selectively ignored.

The simplicity of the EER model and its (almost) direct mapping into a DL can contribute to hide a major issue: the gap resulting from the intrinsic difference between a conceptual knowledge model and an epistemological one. Roughly speaking, there are three main reasons for this gap. Firstly, differences in the modelling abstractions (e.g. entities and relationships in conceptual modelling vs. generic concepts in epistemological modelling). Secondly, different basic assumptions (e.g. closed world vs. open world) underpinning the two knowledge models. Thirdly, different deductions (e.g. hierarchical links between relationships as derived from their $\mathcal{SHIQ}$-based representation and as expected in conceptual reasoning).

Unfortunately, there are no inferential engines available for most conceptual modelling languages, which can provide a set of deductive services comparable with those offered by DL systems. This is often a consequence of the partially undefined semantics of these languages. Therefore, logic-based engines are generally used to perform deductions on conceptual knowledge directly. However, these deductions can be consistently contrasting with the one expected at the conceptual level. Consequently, there is a deductive gap which must be overridden by way of a semantically consistent mapping of the two levels.

The management of this mapping provides a rationale for the introduction of two of the three knowledge representations in CONCEPTOOL, namely the user-oriented (EER) one, the internal (object-oriented) one and the DL-oriented (iFaCT) one. The rationale for an internal knowledge representation can be explained as follows. Several deductions (e.g. completion) do not really need to be performed by the DL inference layer. Conversely, they can be "syntactically" performed, provided that an "augmented" KB can be developed and used to represent, for instance, direct descendants, assertions and propagated knowledge. In this way, overloading of the DL engine is avoided, which results in a better system performance.

Acknowledgements

This work is supported by the EPSRC under grant GR/R10127/01 and under the AKT IRC grant GR/N15764.

References

AKT Consortium. The AKT Manifesto, September 2001. Available from http://www.aktors.org/publications.

E. Compatangelo and D.H. Sleeman. K–ShaRe: Knowledge Sharing and Reuse in AKT. Internal report, Dept. of Computing Science, University of Aberdeen, UK, November 2001.

E. Franconi and G. Ng. The i•com Tool for Intelligent Conceptual Modelling. In *Proc. of the 7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00)*, 2000. Available from http://ceur-ws.org/Vol-29/.

I. Horrocks. FaCT and iFaCT. In *Proc. of the Intl. Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

A hypersequent calculus for Abelian Logic

George Metcalfe
Department of Computer Science
King's College London,Strand, London WC2R 2LS
Email: metcalfe@dcs.kcl.ac.uk

1 Introduction

Abelian logic **A** was introduced independently by Meyer and Slaney in [2] and Casari in [3] as the logic of *lattice-ordered abelian groups*, motivated by the former as a logic of *relevance* and by the latter as a logic of *comparison*. Recently, extending a result in [2], it has been shown that the so-called material fragment of **A** coincides with Lukasiewicz's infinite-valued logic **Ł** [5]. Here we present a cut-free hypersequent calculus for **A**, employing rules familiar from Avron's calculi for Gödel-Dummett logic **LC** and the relevance logic **RM** [1]. Since **Ł** is a fragment of **A** we are also able to fill a hole in the literature identified in [4] and give a hypersequent calculus for **Ł**.

2 Abelian Logic

Definition 1 (Abelian Logic, A) $\rightarrow, +, \wedge, \vee$ *and* t *are primitive. A has the following definitions, axioms and rules:*

$$D\leftrightarrow \quad A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$$
$$D\neg \quad \neg A = A \rightarrow t$$

$$A1 \quad A \leftrightarrow (t \rightarrow A)$$
$$A2 \quad ((A \vee B) \rightarrow C) \leftrightarrow ((A \rightarrow C) \wedge (B \rightarrow C))$$
$$A3 \quad ((A + B) \rightarrow C) \leftrightarrow (A \rightarrow (B \rightarrow C))$$
$$A4 \quad A \rightarrow ((A \rightarrow B) \rightarrow B)$$
$$A5 \quad (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$$
$$A6 \quad A \wedge B \rightarrow A$$
$$A7 \quad A \wedge B \rightarrow B$$
$$A8 \quad ((A \rightarrow B) \wedge (A \rightarrow C)) \rightarrow (A \rightarrow (B \wedge C))$$
$$A9 \quad (A \wedge (B \vee C)) \rightarrow ((A \wedge B) \vee (A \wedge C))$$
$$A10 \quad ((A \rightarrow B) \rightarrow B) \rightarrow A$$

$$(mp) \quad \frac{A \rightarrow B, A}{B} \qquad (\wedge I) \quad \frac{A, B}{A \wedge B}$$

Definition 2 (Lattice-Ordered Abelian Group) *A lattice-ordered abelian group (abelian l-group) is an algebra* $\langle G, +, \vee, \neg, t\rangle$ *with binary operations* $+$ *and* $\vee$*, a unary operation* $\neg$ *and a constant* t*, satisfying the following equations:*

$$a1 \quad t + a = a$$
$$a2 \quad a + b = b + a$$
$$a3 \quad (a + b) + c = a + (b + c)$$
$$a4 \quad a + \neg a = t$$
$$a5 \quad a \vee b = b \vee a$$
$$a6 \quad (a \vee b) \vee c = a \vee (b \vee c)$$
$$a7 \quad a = a \vee a$$
$$a8 \quad a + (b \vee c) = (a + b) \vee (a + c)$$

We also define $a \wedge b = \neg(\neg a \vee \neg b)$, $a \rightarrow b = \neg(a + \neg b)$ *and* $a \leftrightarrow b = (a \rightarrow b) \wedge (b \rightarrow a)$

Theorem 1 (Characterisation theorem for A) *The following are equivalent: (1)* ϕ *is a theorem of A. (2)* ϕ *is valid in all abelian l-groups. (3)* ϕ *is valid in* $(\mathbb{Q}, +, \vee, -, 0)$.

3 A hypersequent calculus

Definition 3 (Hypersequent) *A hypersequent is a multiset of sequents, written:* $\Gamma_1 \vdash \Delta_1 | \ldots | \Gamma_n \vdash \Delta_n$ *where* Γ_i *and* Δ_i *are finite multisets of formulae for* $i = 1, \ldots, n$.

Definition 4 (Interpretation) *The interpretation of a sequent* $S = A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *is* $\phi_S =$

$(A_1+\ldots+A_n) \to (B_1+\ldots B_m)$. *The interpretation of a hypersequent* $H = S_1|\ldots|S_n$ *is* $\phi^H = \phi^{S_1} \vee \ldots \vee \phi^{S_n}$.

Definition 5 (LA: A hypersequent calculus for A) ***LA*** *has the following rules:*

Axioms

$$(id) \quad A \vdash A \qquad (emp) \quad \vdash$$

Structural rules

$$(EW) \quad \frac{G|\Gamma \vdash \Delta}{G|\Gamma \vdash \Delta|\Gamma' \vdash \Delta'} \qquad (EC) \quad \frac{G|\Gamma \vdash \Delta|\Gamma \vdash \Delta}{G|\Gamma \vdash \Delta}$$

$$(S_c) \quad \frac{G|\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}{G|\Gamma_1 \vdash \Delta_1|\Gamma_2 \vdash \Delta_2} \qquad (M) \quad \frac{G|\Gamma_1 \vdash \Delta_1 \quad G|\Gamma_2 \vdash \Delta_2}{G|\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

Logical rules

$$(\to, l) \quad \frac{G|\Gamma, B \vdash A, \Delta}{G|\Gamma, A \to B \vdash \Delta} \qquad (\to, r) \quad \frac{G|\Gamma, A \vdash B, \Delta}{G|\Gamma \vdash A \to B, \Delta}$$

$$(+, l) \quad \frac{G|\Gamma, A, B \vdash \Delta}{G|\Gamma, A + B \vdash \Delta} \qquad (+, r) \quad \frac{G|\Gamma \vdash A, B, \Delta}{G|\Gamma \vdash A + B, \Delta}$$

$$(\wedge_i, l) \quad \frac{G|\Gamma, A_i \vdash \Delta}{G|\Gamma, A_1 \wedge A_2 \vdash \Delta} \qquad (\wedge, r) \quad \frac{G|\Gamma \vdash A, \Delta \quad G|\Gamma \vdash B, \Delta}{G|\Gamma \vdash A \wedge B, \Delta}$$

$$(\vee, l) \quad \frac{G|\Gamma, A \vdash \Delta \quad G|\Gamma, B \vdash \Delta}{G|\Gamma, A \vee B \vdash \Delta} \qquad (\vee_i, r) \quad \frac{G|\Gamma \vdash A_i, \Delta}{G|\Gamma \vdash A_1 \vee A_2 \vdash \Delta}$$

Theorem 2 ***LA*** *is sound and complete for* ***A*** *ie for a hypersequent* H, H *is provable in* ***LA*** *iff* ϕ^H *is valid in all abelian l-groups.*

References

[1] The method of hypersequents in the proof theory of propositional nonclassical logics. A. Avron. In European Logic Colloquium,W. Hodges et al. Ed., Oxford, Oxford Science Publications, Clarendon Press, 1996.

[2] Abelian Logic (from A to Z). R. K. Meyer and K. Slaney. In Paraconsistent Logic Essays on the Inconsistent, G. Priest et al. Ed., Philosophia Verlag, 1989.

[3] Comparative Logics and Abelian l-Groups. E. Casari. In Logic Colloquium '88, Ferro et al. Ed., Elsevier, 1989.

[4] Proof theory of fuzzy logics: Urquart's C and related logics. M. Baaz, A. Ciabatonni, C. Fermuller and H. Veith. In Mathematical Foundations of Computer Science (MFCS98). Lecture Notes in Computer Science, vol. 1450, pages 203-212, 1998.

[5] Analytic sequent calculi for abelian and Łukasiewicz logics. G. Metcalfe, N. Olivetti and D. Gabbay. In preparation.

Resolution for Temporal Logics of Knowledge with Interactions

Cláudia Nalon, Clare Dixon and Michael Fisher
Department of Computer Science
University of Liverpool, Liverpool, L69 7ZF, UK.
E-mail: {C.Nalon, C.Dixon, M.Fisher}@csc.liv.ac.uk

Combined temporal and modal logics have been used in computer science for reasoning about complex situations such as, for instance, the specification and verification of distributed [5] and multi-agent systems [9]. We consider a particular combination, the fusion of PTL and $\mathsf{S5}_{(n)}$, a *Temporal Logic of Knowledge* ($\mathsf{KL}_{(n)}$) in which the dynamic and informational components of such systems can be represented. Given a logical characterisation of a system, we then want to *prove* that it matches the desired requirements and properties. Although there are proof methods for such logics [5, 4, 3, 7], most of these cannot be easily adapted to deal with interactions between the temporal and epistemic dimensions.

We say that time and knowledge interact when axioms relating the temporal and epistemic dimensions are added to the logic. For instance, the axioms $\mathrm{K}_i \bigcirc\varphi \Rightarrow \bigcirc \mathrm{K}_i \varphi$ (*perfect recall and synchrony*) and $\bigcirc \mathrm{K}_i \varphi \Rightarrow \mathrm{K}_i \bigcirc\varphi$ (*no learning and synchrony*) [5] express how the knowledge of an agent evolves over time: the first models situations at which agents remember all their past history, whilst the second models situations at which agents do not distinguish their futures. Adding interacting axioms usually increases the complexity of the validity problem for the combined logic. For instance, on the single-agent case, the validity problem for the combined logic without interactions, $\mathsf{KL}_{(1)}$, is PSPACE, whereas if, in particular, the *no learning and synchrony* interaction axiom is added, the complexity becomes EXPSPACE [6].

Recently, it has been shown that the resolution method for $\mathsf{KL}_{(n)}$ proposed in [3] can also be applied, with few modifications, to deal with perfect recall and synchrony [2] and with no learning and synchrony [8]. Instead of adding new resolution rules, which would make implementation more difficult, we introduce a set of clauses, which make explicit the constraints imposed by the interaction axioms. Also, a number of definitions have to be added to the set of clauses before the resolution method is performed. Many of these definitions are used for renaming complex formula in the scope of the knowledge operator (K_i).

We have now proposed a new normal form, which avoids the introduction of these definitions for complex formulae, and so reducing the set of clauses to which the resolution method is applied. This new normal form was inspired by that proposed in [1], which simplifies the resolution method for temporal logics. We then benefit from this improvement in the method for dealing with the temporal dimension, and have adapted the resolution rules for the knowledge dimension. The resulting proof method for $\mathsf{KL}_{(n)}$ is terminating, sound and complete. Proving correctness of the extended method for synchronous systems with perfect recall or no learning is ongoing work.

We are also investigating how to extend the basic method to deal with interactions of the form $\bigcirc^l \mathrm{K}_i \phi \Rightarrow \mathrm{K}_i \bigcirc^r \phi$ and $\mathrm{K}_i \bigcirc^l \phi \Rightarrow \bigcirc^r \mathrm{K}_i \phi$, where $l, r \in \mathbf{N}$, $\bigcirc^0 \phi = \phi$, and $\bigcirc^i \phi = \bigcirc \bigcirc^{i-1} \phi$ for $i > 0$. Whilst, in general, interactions of these forms have not been studied in computer science, some of their instances, such as the above-mentioned axioms, are very well known, as they express important properties of multi-agent systems. This is the first step towards a general method for dealing with general interactions between modalities representing different dimensions of reasoning.

References

[1] A. Degtyarev and M. Fisher. Towards first-order temporal resolution. In *Proceedings of KI-2001*, volume 2174 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[2] C. Dixon and M. Fisher. Clausal Resolution for Logics of Time and Knowledge with Synchrony and Perfect Recall. In *Proceedings of ICTL 2000*, Leipzig, Germany, 2000.

[3] C. Dixon and M. Fisher. Resolution-Based Proof for Multi-Modal Temporal Logics of Knowledge. In S. Goodwin and A. Trudel, editors, *Proceedings of the Seventh International Workshop on Temporal Representation and reasoning (TIME'00)*, pages 69–78, Cape Breton, Nova Scotia, Canada, July 2000. IEEE Computer Society Press.

[4] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[6] J. Y. Halpern and M. Y. Vardi. The Complexity of Reasoning about Knowledge and Time: Synchronous Systems. Technical Report RJ 6097, IBM Almaden Research Center, San Jose, California, April 1988.

[7] U. Hustadt, C. Dixon, R.A. Schmidt, and M. Fisher. Normal Forms and Proofs in Combined Modal and Temporal Logics. In *Proceedings of the Third International Workshop on Frontiers of Combining Systems (FroCoS'2000)*, volume 1794 of *Lecture Notes in Artificial Intelligence*, pages 73–87. Springer-Verlag, 2000.

[8] C. Nalon, C. Dixon, and M. Fisher. Resolution for Synchrony and No Learning: Preliminary Report. In *Proceeedings of the 8th Workshop on Logic, Language, Information, and Computation - WoLLIC'2001*, Logic Journal of the IGPL, 2001.

[9] A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–342, 1998.

Lakatos-style Reasoning

Alison Pease, Simon Colton, Alan Smaill, John Lee
Division of Informatics, University of Edinburgh,
80 South Bridge, Edinburgh, EH1 1HN,UK
{alisonp@dai., simonco@dai.,A.Smaill@,john@cogsci.}ed.ac.uk

A problem in modelling mathematics is that few people have analysed and reported the way in which mathematicians work. Lakatos(4) is a welcome exception. He presents a rational reconstruction of the evolution of Euler's conjecture that for all polyhedra, the number of vertices (V) minus the number of edges (E) plus the number of faces (F) is two, and its proof. This work spans 200 years of concepts, conjectures, counter-examples and 'proofs' and is invaluable to AI researchers trying to model mathematical reasoning. Such models might serve to *(a)* illuminate aspects of human mathematics, and *(b)* improve existing automated reasoning programs.

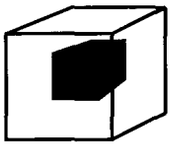

Figure 1: The hollow cube; $V - E + F = 16 - 24 + 12 = 4$

Lakatos suggests that Euler's conjecture may have been proposed after noticing that the relationship $V-E+F = 2$ holds for all regular polyhedra (the tetrahedron, cube, octahedron, icosahedron and dodecahedron). The counter-example of the hollow cube - a cube with a cube-shaped hole in it - is then found. Reactions to this include:

1) saying that the existence of the counter-example is sufficient to disprove the conjecture, and therefore abandon it;

2) claiming that the hollow cube is *not* a counter-example since it is not a polyhedron. The conjecture remains unchallenged and the definition of polyhedron is tightened from 'a solid whose surface consists of polygonal faces' to 'a surface consisting of a system of polygons' (thus excluding the hollow cube);

3) generalising from the hollow cube to 'any polyhedra with a cavity' and then excluding these from the conjecture. The new conjecture is 'for all polyhedra *without cavities*, $V - E + F = 2$';

4) generalising from regular polyhedra to 'convex polyhedra' and then limiting the conjecture to these. The new conjecture is 'for all *convex* polyhedra, $V - E + F = 2$'.

Lakatos categorises these responses into methods: *induction* (scientific rather than mathematical) for generating the initial conjecture; and then if a counter-example is found: 1) *surrender*; 2) *monster-barring*; and two types of *exception-barring* - 3) *piecemeal exclusion* and 4)*strategic withdrawal.* Our aim is to model these methods, using the theory formation program HR(2) as a base. HR starts with objects of interest (*e.g.*, integers) and initial concepts (*e.g.*, multiplication and addition) and uses production rules to transform old concepts into new ones. For example a production rule might take the concept 'number of divisors of an integer', and change it to the concept 'number of divisors of an integer = 2'. It would then list all integers which share this property (*i.e.*, all the primes in its database). Conjectures, such as concept X implies concept Y, are made empirically by comparing the example sets of different concepts. For instance HR has made the conjecture that if the sum of the divisors of n is prime, then the number of divisors of n is prime (2). Since HR works especially well in number theory we have applied the methods to this domain. *Induction* can be used to generate conjectures of the form $\forall x$, $P(x)$; and $-\exists x$ such that $P(x)$, for some property P. Fermat's Last Theorem is an example of this kind of conjecture. Clearly *surrender* is sometimes necessary (many of the conjectures and concepts generated will not be worth modifying) although few surrendered conjectures are recorded. One example in (1) is the conjecture that the *nth* perfect number P_n contains n digits. This was found by induction on the first four perfect numbers - 6, 28, 496, 8128 - but surrendered when the fifth - 33,550,336 -

was found. An example of *monster-barring* is in (3), in which the definition of a prime number is modified from 'a natural number that is only divisible by itself and 1' to 'a natural number with exactly two divisors'. This occurs because the number 1 (considered prime in the first but not in the second definition) is a counter-example to many conjectures. For example in the Fundamental Theorem of Arithmetic, that every natural number is either prime or can be expressed uniquely as a product of primes, the uniqueness criterion is violated ($6 = 2 \times 3 = 2 \times 3 \times 1 = 2 \times 3 \times 1 \times 1$ etc). Rather than explicitly exclude 1 in the theorem, it is more elegant to exclude it from the concept definition, and thus the current definition of a prime is formed and accepted. Examples of *exception-barring* are plentiful, for instance Goldbach's conjecture that all even numbers *except 2* can be expressed as the sum of two primes; all primes *except 2* are odd; and all integers *except squares* have an even number of divisors.

So far we have modelled simple versions of both types of exception-barring in HR. To do this we have had to enable it to generate conjectures with known counter-examples (whereas previously only conjectures true of *all* examples were made). We have done this in two ways (and are currently experimenting to see which is preferable). Firstly we have enabled it to make *near equivalence* conjectures, *i.e.*, conjectures which hold for $x\%$ of the objects of interest (where x is defined by the user). Secondly we have run two versions of HR, which are able to communicate concepts and conjectures to each other, in parallel. These simple agents have access to different databases and therefore one may make a conjecture which is true of all its examples and ask the other for counter-examples. If there is a large number of counter-examples then HR will look to see if it already has a concept which includes only the counter-examples, and then use *piecemeal exclusion.* For instance with a database of numbers $1 - 30$ it formed the near equivalence conjecture that 'all integers have an even number of divisors', with counter-examples 1, 4, 9, 16 and 25. It then found the concept of 'square numbers' which covers the counter-examples and modified the conjecture to 'all integers except squares have an even number of divisors', which is true. If HR makes a conjecture for which there are only a few counter-examples (where the number is specified by the user), it will invent a concept with a definition which excludes the counter-examples and then perform *strategic withdrawal.* When we ran the agent version with databases $1 - 20$ and $20 - 40$ the second agent generated the conjecture 'all primes numbers are odd' and sent it to the first, which found the counter-example 2. The second agent then generated the new concept 'primes numbers except 2' and modified the conjecture to 'all primes numbers except 2 are odd'.

These preliminary results look promising. Although other systems can be said to use certain methods (HR(2) already used *induction* and Lenat's AM program(5) performed a kind of *monster-barring*), this is the only dedicated attempt we know of to model Lakatos-style reasoning.

References

[1] D. Burton. *The History of Mathematics.* Allyn and Bacon, Inc, Boston, USA, 1985.

[2] S. Colton. *Automated Theory Formation in Pure Mathematics.* PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh, 2001.

[3] C. Dunmore. Meta-level revolutions in mathematics. In D. Gillies, editor, *Revolutions in Mathematics*, pages 209–225. Clarendon Press, Oxford, 1992.

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

[5] D. Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics.* PhD thesis, Stanford University, 1976.

Acknowledgements

This work was supported by EPSRC grants GR/M45030. and GR/M98012. The second author is also affiliated with the Department of Computer Science, University of York.

Interactive Proving in *Theorema*

Florina PIROI and Tudor JEBELEAN (*)

Research Institute for Symbolic Computation
A-4232 Hagenberg, Austria
{Florina.Piroi, Tudor.Jebelean}@risc.uni-linz.ac.at

The *Theorema* system is a software intended to assist the working mathematician during all phases of its mathematical work (proving as well), within one consistent logic and one coherent software system. The system is built upon the computer algebra software *Mathematica* [Wol96] which offers unique facilities for the input/output of logical expressions (including complex graphics), for programming by rewrite rules, and for interactivity. In order to make proofs easy to read and interact with, the provers of *Theorema* follow a *natural style* approach: the inferences imitate the natural steps used by human provers, and the rendering of the proofs is done in natural language. This approach has been pioneered by Buchberger, (see e.g. [Bu96], [BuJeVa98]) who designed and implemented the first versions of the predicate logic prover.

The *Theorema* system contains various provers for general and specific domains: a propositional and a predicate logic prover [BuEAl00], the Prove-Solve-Compute (PCS) prover for predicate logic with equality [VaPhD00], induction provers over natural numbers and over lists [BuVa97], a set-theory prover [Wind01], a Groebner Bases based prover for boolean combinations of polynomial equalities and inequalities, etc.

The provers operate on "proof-situations" consisting of a set of assumptions and a goal (sequents with only one goal). Each prover is composed of a set of inference rules, each rule is typically expressed as a rewrite rule which transforms a proof-situation into one or more proof-situations. The proof-object is a tree representation of the development of the proof: each node corresponds to a proof-situation, while its successors in the tree correspond to the inferred proof-situations. The root of the proof-object corresponds to the initial proof-situation, while the leaves correspond to "proved" (successfully solved), "disproved", "failed" or "pending" (not yet resolved) proof-situations.

The proof-object is used as an internal representation of the proof. The external representation of the proof (which is displayed to the user in a proof-notebook) is produced by a post-processing of the proof-object, which generates a structured cell representation in a *Mathematica* notebook. The structure of the cells reflects the tree structure of the proof-object, and each inference step is typically represented as a phrase in natural language accompanied by some formulae and referring to some other formulae by their labels

Normally, a proof call will try to solve/prove the goal by applying the inferences and the heuristics implemented in the prover (or the combination of provers) which is used. Even for powerful proving methods, many proof-problems arising in the current practice of mathematicians are too difficult to solve, mostly because of the large search involved, but also because the impossibility to anticipate in advance all the necessary facts which the formal proof will need as assumptions. In human proving practice, we usually try various possibilities and rely on our intuition and previous experience in order to find an elegant proof.

One of the advantages of natural-style proving is that the user can easily understand and follow both the inference rules which are available in the prover, as well as the proof situation at any given moment. We capitalize on this advantage and, in order to improve the effectiveness of the *Theorema* provers, we implemented a general mechanism which allows a flexible interaction between the user and the system during the development of the proof. Using this, one can choose between fully automatic (for a predetermined number of steps) and interactive proof-development, one can easily navigate through the proof-object, and one can provide various hints (e.g. suitable instantiations) to the prover.

When calling a prover in interactive mode, the system displays the proof in a special *proof-window*, and shows additionally a *proof-situation-window*, a *log-window* and several *Mathematica* style *menu-palettes*.

(*) Partially supported by the RISC phd scholarship program of the government of the Upper Austria and by the FWF (Austrian Science Foundation) SFB project P1302.

The proof-window displays the current situation of the (possibly unfinished) proof. This is in fact the user-friendly representation of the proof-object, and allows the user to navigate inside the proof-tree in order to continue the proof on a certain branch, introduce new branches, etc.

The proof-situation-window displays the proof situation (current goal and assumptions) associated to the node which is currently selected in the proof-tree. This window gives the user a good overview of the current proof situation, and also allows him to indicate the goal or assumption which is to be manipulated in the next proof step (e.g. for applying a certain inference). Pending (still unresolved) proof situations are shown with gray background, and the result of the most recent inference is shown in framed cells.

The log-window logs the main commands of the user and displays the errors and the warnings.

The menu-palettes contain various interactive commands and settings, each being represented by a button which triggers the respective action. Some of the palettes are general, some of them are specific to the particular prover(s) which is/are used. Some of the commands require arguments (e.g. Execute needs a Prove command, NewGoal needs a formula). In order to correctly use such a command, the user selects first the cell containing the needed information.

In very concise words the main menu-palettes are: the **Prove** palette, containing the main commands for controlling the development of the proof (start/finish execution etc.); the **Proof** palette which contains commands that allow the modification of a proof situation of a certain proof step; the **Simplification** palette that provides "shortcuts" to various proof simplification operations which are mostly used; the **Provers** palette which allows the user to select the domain-specific prover which will be used for the next inferences.

The implementation of the palettes is done in such a way that it is easy to add new palettes and new buttons, and also to arrange them in various ways on the user-screen. We plan to enhance these menus according to the results of the experiments with the users. In particular, we plan to add a special dynamic palette which is shows suggestions for the next inference step.

The work presented here has an experimental flavour: one has to try out various types of simplification and interaction schemes in a real environment in order to take the appropriate decisions, and one has to implement the system in a flexible way in order to be able to easily implement the changes suggested by practical experiments with the users (currently, besides the members of the *Theorema* group, we also use as alpha–testers the PhD students enrolled in a special proof-training program at our institute). The presentation will be accompanied by a live demo of the system.

References

[Bu96] B. Buchberger. Using Mathematica for Doing Simple Mathematical Proofs. InProceedings of the 4th Mathematica Users' Conference, Tokyo, November 2, 1996, pages 80-96, Wolfram Media Publishing, 1996.

[BuEAl00] B. Buchberger and C. Duprè and T. Jebelean and K. Kriftner and K. Nakagawa and D. Vasaru and W. Windsteiger. The Theorema Project: A Progress Report.In M.Kerber and M.Kohlhase, editor, Proceedings of Calculemus 2000: Integration of Symbolic Computation and Mechanized Reasoning,A.K.Peters,Natick,Massatchussets, 2000.

[BuJeVa98] B. Buchberger and T. Jebelean and D. Vasaru. *Theorema:* an Integrated System form Computation and Deduction in Natural Style. In Proceedings of CADE 98 (Satellite workshop on integration of computing and proving).

[BuVa97] B. Buchberger and D. Vasaru. An Induction Prover for Equalities over Lists. In *Proceedings of the First International Theorema Workshop, Hagenberg, Austria, June 9–10,1997*, RISC–Reportsseries No. 97–20,1997.

[VaPhD00] D.Vasaru–Duprè. Automated Theorem Proving by Integrating Proving, Solving and Computing. The Research Institute for Symbolic Computation, Johannes Kepler University, May 2000. RISC report 00–19.

[Wind01] W. Windsteiger. A Set Theory Prover in Theorema: Implementation and Practical Applications. The Research Institute for Symbolic Computation, Johannes Kepler University, May 2001. RISC report 01–07.

[Wol96] S.Wolfram.The Mathematica Book, Wolfram Media and Cambridge University Press, 1996.

Using a chart to record intermediate results in a model generation theorem prover

Allan Ramsay
Dept of Computation, UMIST,
PO Box 88, Manchester M60 1QD, UK
Allan.Ramsay@umist.ac.uk

Abstract

Model generation [Manthey and Bry, 1988] provides a very straightforward basis on which to construct more elaborate theorem provers where some rules are applied backwards and some forwards. Unfortunately the basic algorithm can lead to a considerable amount of repeated and redundant work. The current paper exploits ideas from chart parsing [Kay, 1973, Kaplan, 1973] to minimise this problem.

1 Background

Various tasks in natural language processing require you to be able to reason: if you want to do anything as a consequence of hearing/reading an utterance/text, you have to be able to work out its consequences[Appelt, 1985, Asher and Lascarides, 1995, Baumgartner and Kühn, 2000, Blackburn et al., 1997, Crouch et al., 2001, Gardent and Konrad, 2000, Hirst, 1987, Hobbs et al., 1993, McRoy and Hirst, 1995, Schubert and Pelletier, 1984, Sperber and Wilson, 1986, Wedekind, 1996]. The situation is complicated by the fact that this reasoning often involves intensional axioms and logical forms. This happens when reasoning about lexical items that explicitly express propositional attitudes (words like *'want'*, *'regret'*, *'expect'*); for dealing with discourse relations such as the theme:rheme distinction or the significance of phonological marks such as contrastive stress; and for providing an account of terms like *'only'* or *'even'* which express an attitude to parts of the utterance [Pulman, 1993]. We have exploited the distinction between backward- and forward-rules that is implicit in Manthey and Bry [1988]'s MODEL GENERATION theorem prover to develop an inference engine for a highly intensional logic, namely a constructive version of Turner [1987]'s PROPERTY THEORY. This language is more expressive than a logic like λProlog [Nadathur and Miller, 1998] in that it allows for NON-UNIFORM PROOFS [Miller et al., 1991, Loveland and Nadathur, 1998]: the price we pay is that inference is even more difficult than usual.

The version of this inference engine reported in [Ramsay, 2001] incorporates a number of optimisations, mostly techniques that have been developed for standard first-order inference and adapted to work with the intensional language we are using. This theorem prover has been used to solve a number of problems in NLP [Ramsay, 1999a,b,c, 2000, Ramsay and Seville, 2000b,a, 2001b,a, Seville and Ramsay, 2000, Gaylard and Ramsay, 2001], but the basic model generation algorithm can lead to a considerable amount of repeated and redundant work. We therefore propose to exploit ideas from chart parsing [Kay, 1973, Kaplan, 1973] to minimise this problem.

2 Chart parsing

Chart parsing was introduced to minimise the extent to which partial and failed analyses of fragments of an utterance are repeatedly investigated during the process of obtaining a global structural analysis of an input sentence. The key observation behind chart parsing is that most parsing algorithms contain a step in which an incomplete analysis is extended by incorporating a new fragment. The so-called FUNDAMENTAL RULE OF CHART PARSING says that if you have an incomplete constituent $\mathtt{X/[Y_1, Y_2 \ldots, Y_n]}$ adjacent to an item of type $\mathtt{Y_1}$ then you can merge them to form a new, more complete item $\mathtt{X/[Y_2, \ldots, Y_n]}$[1]. The crucial point is that if you retain *every* such object that is created during the course of parsing, and use this to block the creation of new but repetitive items, then you can have a cache of positive and negative lemmas which will enable you to avoid carrying out repeated work.

3 Chart inference

The fundamental rule of chart parsing looks extraordinarily like an application of resolution. It should therefore be possible to exploit the same mechanism in a theorem prover. The question is whether the overheads associated with keeping such a collection of lemmas pays for itself in terms of cutting down the search space. In the long run it must do: the argument that chart parsing provides an N^3 algorithm for analysing text in terms of a context-free, i.e. propositional, grammar whereas simple-minded chronological backtracking is exponential has to be adapted when the framework is first-order or worse, but the argument still applies. Thus for the backward-chaining part of the model generation algorithm, which applies solely to the Horn subset of the problem, there will be some performance gains from using a chart.

But for model generation the improvement should be even more dramatic than that. One of the problems with model generation is that it repeatedly generates new literals to be added to the set of Horn sequents and then tries to (a) prove $\bot$, and (b) if that fails then find a new literal to add. This, in the standard presentation, involves attempting the same proofs over and over again. By using a chart the addition of a new literal directly restart suspended, but failed, proofs of $\bot$ and of the availability of forward rules. We no longer have to repeat old proofs: we resume partial proofs immediately at the point at which they were suspended. This will also work for proofs which were suspended because two items failed to unify when further analysis reveals that the items in question are equal. The treatment of equality within the chart-based approach is more awkward than it was in the standard implementation, where we carry the current equivalence sets around in the LABEL [Gabbay, 1996] associated with the proof, but at the same time it allows more effective use of discoveries about what is equal to what.

The current state of the implementation of this notion is that we have the chart in place for avoiding repetition in the backward chaining part of the proof procedure, but that the integration with the forward chaining phase is not yet complete.

[1]The notation $\mathtt{X/[Y_1, \ldots, Y_n]}$ is a variant on the way CATEGORIAL GRAMMAR describes an item of type $\mathtt{X}$ which requires to combine with items of type $\mathtt{Y_1, \ldots, Y_n}$ in order to become 'complete' [Steedman, 2000, Wood, 1993]. In parsing, of course, you need directional information about whether $\mathtt{Y_1}$ should be to the right or left of $\mathtt{X/[Y_1, Y_2 \ldots, Y_n]}$, but this doesn't make things much more complex.

References

D Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, 1985.

N Asher and A Lascarides. Lexical disambiguation in a discourse context. *Journal of Semantics*, 1995.

P Baumgartner and M Kühn. Abducing coreference by model construction. *Journal of Language and Computation*, 1(2), 2000.

P Blackburn, J Bos, M Kohlhase, and H de Nivelle. Inference and computational semantics. In H Bunt and E Thijsse, editors, *3rd International Workshop on Computational Semantics*, pages 5–19, University of Tilburg, 1997.

R Crouch, A Frank, and J van Genabith. Linear logic based transfer and structural misalignment. In H C Bunt, I van der Sluis, and E Thijsse, editors, *4th International Workshop on Computational Semantics*, pages 35–49, University of Tilburg, 2001.

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

C Gardent and K Konrad. Interpreting definites using model generation. *Journal of Language and Computation*, 1(2):215–230, 2000.

H Gaylard and A M Ramsay. *Any*: The hearer's role in discourse update. In G Angelova, K Bontcheva, R Mitkov, N Nicolov, and N Nikolov, editors, *Recent Advances in Natural Language Processing*, pages 108–114, Tzigov Chark, Bulgaria, 2001.

G Hirst. *Semantic interpretation and the resolution of ambiguity.* Studies in natural language processing. Cambridge University Press, Cambridge, 1987.

J R Hobbs, M E Stickel, D E Appelt, and P Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

R M Kaplan. A general syntactic processor. In R. Rustin, editor, *Natural language processing*, pages 193–241, New York, 1973. Algorithmics Press.

M Kay. The MIND system. In R. Rustin, editor, *Natural Language Processing*, pages 155–188, New York, 1973. Algorithmics Press.

D W Loveland and G Nadathur. Proof procedures for logic programming. In D M Gabbay, C J Hogger, and J A Robinson, editors, *Handbook of logic in artificial intelligence and logic programming: volume 5, logic programming*, pages 163–234. Clarendon Press, Oxford, 1998.

R Manthey and F Bry. Satchmo: a theorem prover in Prolog. In *Proceedings of the 9th International Conference on Automated Deduction (CADE-9)*, volume 310 of *Lecture Notes in Artificial Intelligence*, pages 415–434, Berlin, 1988. Springer-Verlag.

S W McRoy and G Hirst. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):435–478, 1995.

D Miller, G Nadathur, F Pfennig, and A Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

G Nadathur and D Miller. Higher-order logic programming. In D M Gabbay, C J Hogger, and J A Robinson, editors, *Handbook of logic in artificial intelligence and logic programming: volume 5, logic programming*, pages 499–590. Clarendon Press, Oxford, 1998.

S G Pulman. Higher order unification and the interpretation of focus. *Linguistics & Philosophy*, 20(1):73–115, 1993.

A M Ramsay. Direct parsing with discontinuous phrases. *Natural Language Engineering*, 5(3): 271–300, 1999a.

A M Ramsay. Dynamic and underspecified semantics without dynamic and underspecified logic. In H Bunt, L Kievit, R Muskens, and M Verlinden, editors, *Computing Meaning*, volume 1, pages 208–220, Dordrecht, 1999b. Kluwer Academic Publishers (SLAP 73).

A M Ramsay. Weak lexical semantics and multiple views. In H C Bunt and E G C Thijsse, editors, *3rd International Workshop on Computational Semantics*, pages 205–218, University of Tilburg, 1999c.

A M Ramsay. Speech act theory and epistemic planning. In W J Black and H Bunt, editors, *Abduction, Beliefs and Context in Dialogue: Studies in Computational Pragmatics*, pages 293–310, Amsterdam, 2000. John Benjamins.

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

A M Ramsay and H Seville. Models and discourse models. *Journal of Language and Computation*, 1(2):167–181, 2000a.

A M Ramsay and H Seville. Unscrambling English word order. In M Kay, editor, *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 656–662, Universität des Saarlandes, July 2000b.

A M Ramsay and H Seville. Relevant answers to wh-questions. In *3rd International Conference on Inference in Computational Semantics*, pages 73–86, Siena, 2001a.

A M Ramsay and H Seville. What situational and discourse relations entail. In H C Bunt, I van der Sluis, and E Thijsse, editors, *4th International Workshop on Computational Semantics*, pages 417–432, University of Tilburg, 2001b.

L Schubert and F J Pelletier. From English to logic: Context-free computation of 'conventional' logical translations. *American Journal of Computational Linguistics*, pages 165–176, 1984.

H Seville and A M Ramsay. Making sense of reference to the unfamiliar. In M Kay, editor, *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 775–781, Universität des Saarlandes, July 2000.

D Sperber and D Wilson. *Relevance: communication and cognition.* Blackwell, Oxford, 1986.

M Steedman. *The Syntactic Process.* MIT Press, Cambridge, Ma., 2000.

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

J Wedekind. On inference-based procedures for lexical disambiguation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 980–985, Copenhagen, 1996.

M M Wood. *Categorial Grammars.* Routledge, London, 1993.

BRAIN: Backward Reachability Analysis with INtegers

Tatiana Rybina and Andrei Voronkov

University of Manchester
{rybina,voronkov}@cs.man.ac.uk

1 General description

BRAIN is a tool for analysis of infinite-state systems in which a state can be represented by a vector of integers. Currently it is used to verify safety and deadlock properties expressed as reachability statements in a quantifier-free language with variables ranging over natural numbers.

2 Input language

The input language of BRAIN consists of problem descriptions specifying the following.

1. The sets of initial and final states by a quantifier-free formula of Presburger arithmetic. Instead of final states, one can specify the set of deadlock states, which will be computed by BRAIN automatically from the description of transitions.
2. Transitions expressed as *guarded assignments* `guard -> assignment` (see [2]), where `guard` is again a quantifier-free formula of Presburger arithmetic and `assignment` is a sequence of assignments to variables. Examples will be given below.
3. Queries of two types: reachability of a set of states from another set of states, or reachability of a deadlock state from a set of states.

3 Transition systems

The semantics of a problem expressed in the language can be defined as follows. Let $\mathcal{V}$ be the set of all variables declared in the problem. We call a *state* a mapping from the set of variables into the corresponding domain (currently, only the domain of natural numbers is supported). Such a mapping can be homomorphically extended to arbitrary terms of Presburger arithmetic over $\mathcal{V}$ by defining $s(n) = n$, for every non-negative integer n and $s(u_1 + u_2) = s(u_1) + s(u_2)$. A *guard* G is a quantifier-free formula of Presburger arithmetic with variables in $\mathcal{V}$. In addition to the function symbol $+$ and equality BRAIN allows one to use $-$ and all the standard comparison operators, such as $\leq$. If a formula C of variables $\mathcal{V}$ is true in a state s, we write $s \models C$. Every such formula C *symbolically represents* a set of states, namely the set of states in which it is true, i.e., $\{s \mid s \models C\}$. A *transition* t is a set of pairs of states. Transitions are declared in BRAIN using *guarded assignments* of the form $G \to v_1 := u_1, \ldots, v_n := u_n$, where all the v_i's are variables and the u_i's are terms. This guarded assignment defines a (deterministic) transition t with the following properties: $(s, s') \in t$ if $s \models G$, $s'(v_1) = s(u_1), \ldots, s'(v_n) = s(u_n)$, and for every variable $v \in \mathcal{V} - \{v_1, \ldots, v_n\}$ we have $s'(v) = s(v)$. Though every guarded assignment specifies a deterministic transition, the transition relation specified by a set of guarded assignments may be non-deterministic.

4 The reachability algorithm

The backward reachability algorithm LocalBackward of BRAIN is implemented via a saturation algorithm given in Figure 1. This algorithm is taken from [3], where implementation of BRAIN is discussed in more detail. Backward reachability was chosen for a simple reason: symbolic application of a guarded assignment to a set of states represented by a formula $C(\mathcal{V})$ gives a quantified formula, while quantifiers arising from the backward application of a guarded assignment can be immediately eliminated.

```
procedure LocalBackward
input: quantifier-free formulas In, Fin,
       finite set of simple guarded assignments U
output: "reachable" or "unreachable"
begin
  IS := pdnf(In); FS := pdnf(Fin)
  if there exist I ∈ IS, F ∈ FS such that ⊨ ∃V(I ∧ F) then return "reachable"
  unused := FS; used := ∅
  while unused ≠ ∅
    S := select(unused)
    used := used ∪ {S}; unused := unused − {S}
    forall u ∈ U
      (* backward application of u *)
      N := S^{-u}
      (* satisfiability-check for the new constraint N *)
      if ⊨ ∃V(N) then
        (* intersection-checks *)
        if there exists I ∈ IS such that ⊨ ∃V(N ∧ I) then return "reachable"
        (* entailment-checks *)
        if for all C ∈ used ∪ unused we have ⊭ ∀V(N ⊃ C) then
          unused = unused ∪ {N}
          forall C' ∈ used ∪ unused
            (* more entailment-checks *)
            if ⊨ ∀V(C' ⊃ N) then remove C' from used or unused
  return "unreachable"
end
```

Fig. 1. Local backward reachability algorithm used in BRAIN

5 Algorithms over simple constraints

The backward reachability algorithm of BRAIN requires efficient algorithms for checking satisfiability and entailment of constraints. These algorithms are described in [3]. They are based on an algorithm [4] for building the basis of simple constraints (systems of linear equalities and inequalities over nonnegative integers) similar to the algorithm of [1]. The algorithm computes the basis of a constraint consisting of non-decomposable solutions. Incremental computation of the basis is very useful for checking intersection with the set of the initial states and also for deleting redundant inequalities or equalities.

6 Performance

Paper [3] contains experimental results comparing BRAIN with Action Language Verifier, HyTech and DMC. For integer-valued problems HyTech and DMC use relaxation, i.e., they solve real reachability problems instead of integer reachability problems. Therefore, they are correct only when they report non-reachability. Our results reported in [3] show that BRAIN is usually faster that HyTech and considerably faster than the other two systems, often by several orders of magnitude. BRAIN also consumes much less memory that HyTech and Action Language Verifier. There are several problems which could only be solved by BRAIN, but not by any of the other systems. On some of these benchmarks, BRAIN made about 10^9 entailment checks within the overall running time of about 15 minutes. However, we would like to note that all of these systems are on some benchmarks more powerful than BRAIN since they can use techniques such as widening or transitive closure which the current version of BRAIN does not have. The system is available at `http://www.cs.man.ac.uk/~voronkov/BRAIN/`.

References

1. F. Ajili and E. Contejean. Avoiding slack variables in the solving of linear diophantine equations and inequations. *Theoretical Computer Science*, 173(1):183–208, 1997.
2. T. Rybina and A. Voronkov. A logical reconstruction of reachability. In *Submitted to FME 2002*, 2002.
3. T. Rybina and A. Voronkov. Using canonical representations of solutions to speed up infinite-state model checking. In *Submitted to CAV 2002*, 2002.
4. A. Voronkov. An incremental algorithm for finding the basis of solutions to systems of linear Diophantine equations and inequations. unpublished, January 2002.

Substitutions, test and knowledge in axiomatic products of *PDL* and *S5*

Renate A. Schmidt and Dmitry Tishkovsky

Department of Computer Science,
University of Manchester
Manchester M13 9PL, United Kingdom,
{schmidt,dmitry}@cs.man.ac.uk

Combinations of propositional dynamic logic (*PDL*) [3] and *S5* [1, 5] are meaningful when we want to reason about actions and knowledge. We focus on axiomatically defined products of *PDL* and *S5*, in which the *S5* modality commutes with the dynamic operators by the following axiom schema.

(NL ∧ PR) $\quad [a]\Box p \leftrightarrow \Box[a]p$

(**NL** is short for no learning, and **PR** for perfect recall.) Generally axioms and theorems of a logic are assumed true for all instantiations for the atomic symbols. However, in this paper we distinguish between two variants of the substitution rule. The *weak substitution rule* allows the substitution of arbitrary formulae for the atomic propositional symbols but does not allow substitution for atomic action symbols. By contrast, the *full substitution rule* allows both kinds of substitutions.

Let $\mathcal{L}$ be the language of *PDL* plus an additional modal operator $\Box$ representing knowledge. A *logic* (*weak logic*) in $\mathcal{L}$ is a set of formulae which is closed under standard inference rules (modus ponens and necessity rule) and under the full (resp. weak) substitution rule. Notationally weak logics are discerned by a subscript w.

The simplest form of combination of two (or more) logics is their fusion, or independent join. We denote the *fusion* of *PDL* and *S5* by *PDL*⊕*S5*.

Proposition 1. *$PDL \oplus S5 = (PDL \oplus S5)_w$, i.e. full substitution is admissible in $(PDL \oplus S5)_w$.*

$PDL \oplus S5$-models are combinations of the familiar Kripke models for *PDL* and *S5*, that is, a *$PDL \oplus S5$ model* is a tuple $\langle S, Q, R, \models\rangle$ such that $\langle S, Q, \models\rangle$ is a *PDL*-model [3] and $\langle S, R, \models\rangle$ is an *S5*-model [1].

The products of *PDL* and *S5* we consider are extensions of *(test-free) PDL* ⊕ *S5* with the axiom (**NL** ∧ **PR**) above. This axiom is a Sahlqvist formula and its first-order equivalent is:

(com) $\quad R \circ Q(a) = Q(a) \circ R$

In the case of logics with full substitution a ranges over all the actions, whereas for weak logics a ranges over the set of atomic actions only and in this case we will refer to this property as $\mathbf{com}_w$. We consider the logics:

$$[PDL, S5] \rightleftharpoons PDL \oplus S5 + \mathbf{NL} \wedge \mathbf{PR}$$

$$[\textit{test-free } PDL, S5] \rightleftharpoons \textit{test-free } PDL \oplus S5 + \mathbf{NL} \wedge \mathbf{PR}$$

and their weak versions $[PDL, S5]_w$ and $[\textit{test-free } PDL, S5]_w$. These extensions contain all possible definitions of axiomatic products of *(test-free) PDL* and *S5*. By definition, a $[PDL, S5]$-*model* (resp. $[\textit{test-free } PDL, S5]$-*model*) is a $PDL \oplus S5$-model (resp. test-free $PDL \oplus S5$-model) which satisfies property **com**. We have similar definitions for weak logics with the property $\mathbf{com}_w$ instead of **com**.

The following theorem tells us something about the admissibility of the full substituion rule in weak variants these logics.

Theorem 1. *$[PDL, S5] \neq [PDL, S5]_w$, but $[\textit{test-free } PDL, S5] = [\textit{test-free } PDL, S5]_w$.*

All considered logics can be proved complete, decidable, and have the small model property.

Theorem 2 (Small Model Property). *Let L be any of the logics $[PDL, S5]$, $[PDL, S5]_w$, or $[\textit{test-free } PDL, S5]$. Let ϕ be a formula of $\mathcal{L}$ and n the number of symbols in ϕ. If ϕ is satisfiable in some L-model then it is satisfiable in an L-model with no more than $\mu(L, n)$ states, where $\mu([PDL, S5], n) = 2^n$ and $\mu([PDL, S5]_w, n) = \mu([\textit{test-free } PDL, S5], n) = 2^n \cdot 2^{2^n}$.*

The knowledge modality can be trivially eliminated from $[PDL, S5]$.

Proposition 2. $p \leftrightarrow \Box p \in [PDL, S5]$.

This *trivial* elimination of the *S5* operator in the product $[PDL, S5]$ is unsatisfactory both from a logical perspective and an application-oriented perspective. The reason for the elimination of the

S5 operator is the implicit connection between the test operator and $\Box$ in the commutativity axioms under full substitutivity. A solution we propose here is to define an alternative semantics for test such that in the resulting logic, $\Box$ and test interact in a way so that weak substitutivity implies full substitutivity.

Therefore, we introduce a new operator, denoted by $?\!\!?$, as replacement for the standard test operator. The new operator can be interpreted as an *epistemic test* operator. The intuition of $p?\!\!?$ is an action which can be succesfully accomplished only if p is *known* in the current state. The result of this action is an arbitary state within the same knowledge cluster. Thus, $p?\!\!?$ is the action of confirming the agents's own knowledge. In contrast, with the usual test operator the agent has the capability to confirm truths rather than knowledge. Philosophically, this is a strong property of an agent; we believe, too strong. In agent based applications the new interpretation of test is more suitable than the usual test operator.

Subsequently, the logical apparatus is the same as previously with the obvious changes. The symbol $?\!\!?$ is used in the superscript to indicate a replacement of the operator ? by $?\!\!?$. Let $[PDL, S5]^{?\!\!?}$ and $[PDL, S5]^{?\!\!?}_w$ be the logics in $\mathcal{L}^{?\!\!?}$ obtained from $[PDL, S5]$ and $[PDL, S5]_w$, respectively, by replacing the usual test axiom with:

$$[p?\!\!?]q \leftrightarrow \Box(\Box p \rightarrow q).$$

In accordance with this axiom, the formula $[p?\!\!?]q$ can be read as 'q is known with respect to p being known'. Thus, we think of the modal operator $[\_?\!\!?]$ as the operator of *relative knowledge*.

Using the elimination of second-order quantifiers [2, 4] it is easy to find the corresponding semantic definition for the new operator. Thus, a $(PDL \oplus S5)^{?\!\!?}$ model is a tuple $\langle S, Q, R, \models\rangle$ satisfying all the properties of $PDL \oplus S5$ model, except the meaning of $?\!\!?$ is specified by:

$$Q(\phi?\!\!?) \rightleftharpoons \{(s,t) \in R \mid t \models \Box\phi\}.$$

This induces the notions of $[PDL, S5]^{?\!\!?}_w$ and $[PDL, S5]^{?\!\!?}$ models as expected.

The definition of $?\!\!?$ still allows the elimination of $\Box$ but this time the elimination is not trivial.

Proposition 3. $\Box p \leftrightarrow [\top?\!\!?]p \in PDL \oplus S5^{?\!\!?}$.

In this context, we obtain the theorem of admissibility of the full substitution rule in $[PDL, S5]^{?\!\!?}_w$.

Theorem 3. $[PDL, S5]^{?\!\!?} = [PDL, S5]^{?\!\!?}_w$.

Also the following theorem is proved.

Theorem 4. *$[PDL, S5]^{?\!\!?}$ has small model property with $\mu([PDL, S5]^{?\!\!?}, n) = 2^n \cdot 2^{2^n}$ and is complete and decidable.*

How does the standard test operator of *PDL* relate to the new one? It turns out that there is a simulation of *PDL* in $(PDL \oplus S5)^{?\!\!?}$. Define the translation σ from formulae of *PDL* to $\mathrm{For}^{?\!\!?}$ by the following:

$$\begin{array}{rlrl} \sigma a &= a & \sigma\bot &= \bot \\ \sigma(\psi?) &= (\sigma\psi)?\!\!? & \sigma p &= \Box p \\ \sigma(\alpha \cup \beta) &= \sigma\alpha \cup \sigma\beta & \sigma(\phi \rightarrow \psi) &= \Box(\sigma\phi \rightarrow \sigma\psi) \\ \sigma(\alpha;\beta) &= \sigma\alpha;\sigma\beta & \sigma([\alpha]\psi) &= \Box[\sigma\alpha]\sigma\psi \\ \sigma(\alpha^*) &= (\sigma\alpha)^* & & \end{array}$$

Theorem 5. *For any $\Box$-free formula ϕ in $\mathcal{L}$, $\phi \in PDL$ iff $\sigma\phi \in (PDL \oplus S5)^{?\!\!?}$.*

References

1. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge Univ. Press, 2001.
2. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7:35–43, 1992.
3. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.
4. H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In S. Kanger, editor, *Proc. 3rd Scandinavian Logic Symposium, 1973*, pp. 110–143. North-Holland, 1975.
5. M. Zakharyaschev, F. Wolter, and A. Chagrov. *Advanced Modal Logic.* Kluwer, 1998.

Proof Planning Diagonalisation Theorems via Category Theory

Alan Smaill A.Smaill@ed.ac.uk
Division of Informatics, University of Edinburgh

1 "The" diagonal argument?

Various automated reasoning approaches to diagonalisation arguments have approached the subject via a meta-theoretic account of proof state, or reformulation via syntactic manipulation (*eg* [HKC95, CS98, Mel94]). An alternative approach is based upon a more abstract characterisation of diagonal arguments, where a single mathematical argument at the level of category theory can be instantiated into a wide range of diagonal arguments. Category theory shares with proof planning the desire to generalise patterns of argument, though it goes about this in a different way. The claim here is that this different separation between object-level mathematics and meta-level control gives us more mathematical feel for the shape of the proofs concerned.

Lawvere and Schanuel's book [LS97], which we follow here, is intended as a way into mathematics for "the general reader or beginning student". In it (in section 29) there is a good discussion of diagonalisation, based on Lawvere's earlier research paper [Law69]. I assume basics of category theory in the following summary. Suppose we are in a category with products, *ie* we have:

- Objects $X, Y, T, \ldots$; with an associated notion of
- Maps $f, g \ldots$, between these objects which can be composed in sensible ways (including identity maps from an object to itself); and
- Product objects like $X \times Y$. These are characterised via projection maps, *eg* $\pi_X : X \times Y \to X$ and the ability to form a map $\langle f, g \rangle : A \to X \times Y$ given maps $f : A \to X, g : A \to Y$.

Now the diagonal map $\delta : X \to X$ is just $\langle 1_X, 1_X \rangle$ where 1_X is the identity map on X. We suppose there is an object $\mathbf{1}$ with exactly one map $X \to \mathbf{1}$ for every X. For any $\phi : T \times X \to Y$, there is a family of maps $T \to Y$ indexed by maps $t : \mathbf{1} \to X$. If *all* the maps $T \to Y$ are in this family, we say that $\phi : T \times X \to Y$ *parametrises* the maps $T \to Y$.

From this can be shown both positive and negative forms of the diagonal argument. The latter says that:

> **Diagonal Argument, Contrapositive.** In a category C with products, if there is some $\alpha : Y \to Y$ with no fixed points, then for every T and every purported parametrisation $\phi : T \times T \to Y$, there is a map $f : T \to Y$ which is left out of the parametrised family.

Out of this drops for example that looking at sets and functions, no map $T \times T \to bool$ (where *bool* is simply a two-element set) can parametrise all maps $T \to bool$, since the swap map on *bool* has no fixed-point.

2 Proof planning around this argument

It is plausible that this version of the diagonal argument is sufficiently general that reformulation is not needed, but rather we want to instantiate the ingredients of the argument in a particular context. Typically we need to, in some order:

1. Identify the objects and maps involved.
2. Check that maps compose, that identity map is present.
3. Identify the appropriate product construction, and check its characteristic properties.
4. Look for a fixed-point free map from the appropriate object to itself.

If these all succeed, we have the components for a diagonal argument, in the negative form. If (4) fails, the negative argument fails (with the given choice of objects and maps), though the conclusion may hold anyway.

It is natural to use a higher-order representation in dealing with this sort of problem. An initial implementation in *λClam* shows that this approach is feasible. Notably, it allows us to invent examples (and non-examples) of diagonal arguments outside the standard stock. For example, can we continuously parametrise the set of continuous functions $f : \mathcal{R} \to \mathcal{R}$ by means of some $\phi : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$? We are pointed in the direction of a series of lemmas:

1. Composition of continuous functions are continuous, and the identity map is continuous.
2. The topology on $X \times Y$ is defined so that the usual projections are continuous, and do indeed characterise the product correctly.
3. There are continuous maps $\mathcal{R} \to \mathcal{R}$ with no fixed-points (eg $x \mapsto x + 1$).

And so we conclude that there can be no such continuous ϕ.

[LS97] does a sketch of how incompleteness for arithmetic fits into this set-up. It would be much harder to invent the appropriate constructions here, but not so hard to check that they do fit this pattern.

In conclusion, this different way of organising a family of proofs by seeing them as instances of a single more abstract argument is amenable to implementation in a higher-order framework. It allows us to apply diagonal arguments across a wide range of examples, and gives an explanation for some cases where diagonalisation fails.

References

[CS98] Lassaad Cheikhrouhou and Jörg Siekmann. Planning diagonalization proofs. In *Proceedings of 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, volume 1480 of *Lecture Notes in Artificial Intelligence*, pages 167–180. Springer, 1998.

[HKC95] X. Huang, M. Kerber, and L. Cheikhrouhou. Adapting the diagonalization method by reformulations. In A. Levy and P. Nayak, editors, *Proc. of the Symposium on Abstraction, Reformulation and Approximation (SARA-95)*, pages 78–85. Ville d'Esterel, Canada, 1995.

[Law69] F.W. Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications II*, volume 92 of *Lecture Notes in Mathematics*, pages 143–145. Springer, 1969.

[LS97] F.W. Lawvere and S.H. Schanuel. *Conceptual Mathematics*. Cambridge University Press, Cambridge, England, 1997.

[Mel94] E. Melis. Representing and transferring diagonal methods. Technical Report CMU-CS-94-174, Carnegie Mellon University, 1994.

Using Implicit Induction to Guide a Parallel Search for Inconsistency

Graham Steel, Alan Bundy and Ewen Denney
Division of Informatics
University of Edinburgh
{grahams,bundy,ewd}@dai.ed.ac.uk
http://www.dai.ed.ac.uk/~grahams

February 21, 2002

Abstract

We describe the first full implementation of the Comon-Nieuwenhuis method for implicit induction, including a consistency checker, in a novel system where the proof and refutation programs communicate via sockets. This allows the system to attempt to prove and disprove a conjecture at the same time, using parallel theorem proving processes. As well as refuting several non-theorems, this system has accomplished what is, to the best of our knowledge, the first fully automated proof by implicit induction of the commutativity of *gcd*. This had been posed as a challenge problem to the technique in the past.

1 Overview of The System

Comon and Nieuwenhuis described the operation of a refutation complete proving system based on an implicit induction strategy in their paper, [1], but their implementation was not complete. We have produced a complete implementation, described in figure 1. The input to the system is an inductive problem in `Saturate` format and an I-Axiomatisation (as defined by Comon and Nieuwenhuis, see [1]). The version of `Saturate` customised by Nieuwenhuis for implicit induction (the rightmost box in the diagram) gets the problem file only, and proceeds to pursue inductive completion. Every non-redundant clause generated is passed via the server to the refutation control program (the leftmost box). For every new clause generated, this program generates a problem file containing the I-Axiomatisation and the new clause, and spawns a standard version of `Saturate` to attempt to find an inconsistency in the file. Crucially, these spawned `Saturates` are not given the original axioms. This means that most of the search for an inconsistency is done by the prover customised for inductive problems and the spawned `Saturates` are just used to check for inconsistencies between the spawned clauses and the I-Axiomatisation. This should enable an inconsistency to be found more quickly than if we just gave the axioms, the I-Axioms and the conjecture to a first-order prover.

It is necessary to spawn the standard `Saturate` processes in parallel to the implicit induction process because inductive completion may not terminate. A concurrent check for consistency allows us to detect incorrect conjectures even in this case. For example, the system has refuted $gcd(x, x) = 0$, a problem for which the completion process is non-terminating. If a refutation is found by a spawned prover, the proof is written to a file and the completion process and all the other spawned `Saturate` processes are killed. If completion is reached by the induction prover, this is communicated to the refutation control program, which will then wait for the results from the spawned processes. If they all terminate with saturation, then there are no inconsistencies, and so the theorem has been proved.

All spawned `Saturate` processes output to `stdout`, but a separate `xterm` window is launched for the implicit induction prover. This is because some user interaction is sometimes required for the implicit induction process; if it was connected to the main stream, it would accept the output from another process as its input. The user interaction required is the selection of the innermost defined symbol in order to determine an inductively complete position, [1], taking advantage of the fact that our problems are in constructor theories to reduce the

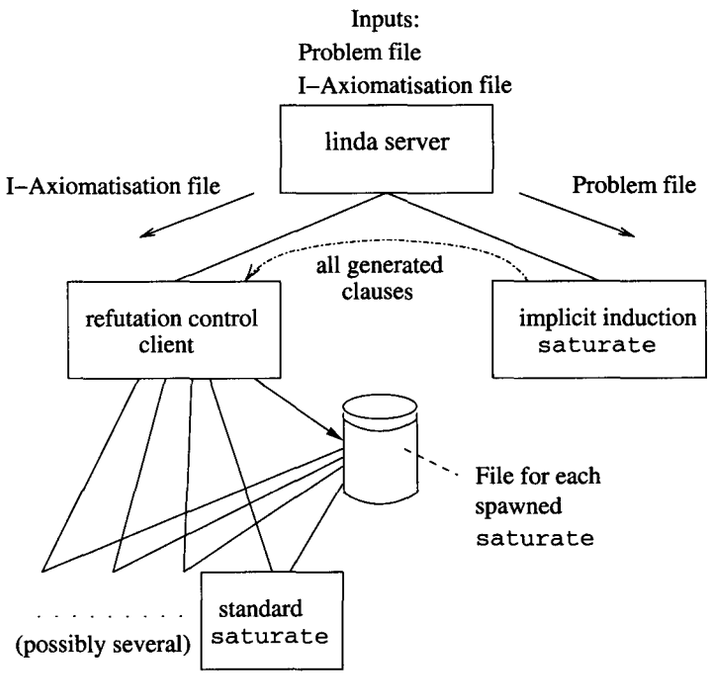

Figure 1: System Operation

amount of search required. This will be automated in the future. When the theorem proving process is over, the user also has to type the commands required to recover the proof.

2 Examples

The system has proved a small number of theorems about the *gcd* function, including commutativity and idempotence, as well as theorems about even numbers and the $>$ predicate. Several non-theorems involving the same function symbols have also been detected. For a table of results and some sample output see `http://www.dai.ed.ac.uk/~grahams/linda/arw.html`.

3 Further Work

Our eventual aim is to use this system (or one of its successors) for investigating conjectures about security protocols, and then to automatically extract attacks from the refutations. The idea is to produce a first-order version of the inductive formalism used by Paulson, [2], allowing us to reason about protocols involving an arbitrary number of agents. The main disadvantage of Paulson's method is that considerable experience with the Isabelle prover is required in order to analyse a new protocol. This is especially true if the proof attempt breaks down, leaving the user unsure as to whether there is a problem with the proof attempt, e.g. another lemma or a generalisation is needed, or whether the protocol itself is faulty. A system which could automatically synthesize attacks in the case of a faulty protocol would alleviate this problem.

References

[1] H. Comon and R. Nieuwenhuis. Induction = I-Axiomatization + First-Order Consistency. *Information and Computation*, 159(1-2):151–186, May/June 2000.

[2] L.C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.

Embedding a quantified logic for spatial reasoning in Isabelle-HOL

Paolo Torrini, Jacques D. Fleuriot,
John G. Stell, Brandon Bennett.
{jgs,brandon,paolot}@comp.leeds.ac.uk (Sc. of Computing, Un. of Leeds)
{jdf,paolot}@dai.ed.ac.uk (Sc. of AI, Un. of Edinburgh)

Spatial semantics have often been discussed, quite recently, both from the perspective of first-order systems [RCC92, PS98] and from that of non-classical propositional logics [Ben96, Lem96, She99, Ste00]. One of the general directions seems to be toward the development of models of space that are closer to common-sense reasoning than the standard one based on a cartesian product of the reals. Qualitative models can either refer to continuous spaces or to digital ones; the latter ones, on which we intend to focus here, can also be intended as multi-resolution cell complexes, and are in general comparatively closer to machine representations [Kov92]. The basic semantical entities are taken to be *regions* that can be topologically interpreted as regular open sets. Some of the spatial relations that can be defined on regions are: parthood, disjointness, disconnection (binary), emptiness, connectedness (unary), and their complements. Topologically, parthood can be interpreted as inclusion, disjointness as empty intersection, disconnection as empty intersection of the respective closures, connectedness as impossibility to be split into two non-empty, disjoint open sets.

It has already been discussed [TB02, TS02], in first place, how the topological semantics for intuitionistic propositional logic (introduced with [Tar56]) can be re-adapted into a semantics for an axiomatic version of intuitionistic second-order propositional logic (ISPL); besides, how an extension of ISPL can be used to encode all the above-mentioned 'positive' relations (and, adding some semantical constraint, also the complementary ones); the encoding does not work for all the spaces, but it does for a subclass of Alexandroff topologies which are close enough to realise an intuitive idea of digital spaces.

Provability in ISPL can be inductively defined in terms of natural deduction, sequent-calculus or Hilbert axiomatisation [Loe76, Gab81]; the definition of formula is of course different from that of intuitionistic first-order predicate calculus (IFPC), since variables have type formula (wff) instead than type term (term), but the postulates are basically the same; the interpretation of quantifiers is substitutional; actually, not all the IFPC postulates are necessary, since all the logical operators are 2nd-order definable from $\rightarrow, \forall$; however, we follow [Gab81] in adding as an extra postulate the following 'comprehension' principle $\vdash \exists x.(x \leftrightarrow A)$

Isabelle is an interactive, generic theorem-prover [Pau94] that has an implementation of higher-order logic (HOL), allowing higher-order syntax for the binders, and making it possible to deal semi-automatically with positive inductive definitions. An embedding of ISPL in Isabelle-HOL can be interesting especially insofar as it allows the mechanisation of inductive meta-proofs; an example is the proof required by the *replacement of equivalents* ($\vdash A \leftrightarrow B$ implies $\vdash C(A/x) \leftrightarrow C(B/x)$), a principle that can be used to simplify quite radically the object-level reasoning.

Nevertheless, there seems to be a significant problem with handling the embedding of ISPL in a direct way; in terms of higher-order syntax, propositional quantification has an 'impredicative' type $(\mathsf{wff} \mapsto \mathsf{wff}) \mapsto \mathsf{wff}$, which is not admissible from the point of view of recursive type declarations for positive inductive definitions. A way to get around this problem appears to be a translation of ISPL into IFPC; the latter can in fact be embedded in Isabelle-HOL in a comparatively standard way, using the rules of sequent calculus. One possibility seems that to rely on a representation with all the rules for IFPC, a one-place predicate *Var* (where $Var(x)$ is intended to be the propositional variable for which the object-variable x stands), and a corresponding version of the comprehension axiom. Another idea is that of using the IFPC rules for $\rightarrow$ and $\forall$, with equality, $\iota : (\mathsf{term} \mapsto \mathsf{wff}) \mapsto \mathsf{term}$ (the description

operator, which can be axiomatised with $\vdash x = \iota y.P(y)$ iff $\vdash P(x) \wedge (\forall y.P(y) \rightarrow y = x))$, *Var*, and a one-place, second-order propositional function which associates a formula A to the object-variable x such that $Var(x) = A$, defined as $Term := \lambda A.\iota x.(Var(x) \leftrightarrow A)$, assuming additionally that ι is defined for every instance of $\lambda x.(Var(x) \leftrightarrow A)$ (with x not free in A), so that $A \leftrightarrow Var(Term A)$ is provable for every A. At the moment of writing, a prototype has been implemented, closer to the second line of thought (though introducing *Term* directly, without ι, and overloading the built-in notion of substitution with a recursive definition that does not affect the treatment of the binders); the automatically generated induction principle provided by Isabelle has been successfully tested on the proof of the replacement of equivalents.

References

[Ben96] B. Bennett. Modal logics for qualitative spatial reasoning. *Bulletin of the Interest Group in Pure and Applied Logic (IGPL)*, 4(1):23–45, 1996.

[Gab81] D.M. Gabbay. *Semantical Investigation in Heyting intuitionistic logic.* Number 148 in Synthese. Reidel, 1981.

[Kov92] V.A. Kovalevsky. Topological foundations of shape analysis. In O. Ying-Lie, A. Toet, D. Foster, H.J.A.M. Heijmans, and P. Meer, editors, *Shape in picture*, pages 21–37. Springer-Verlag, 1992.

[Lem96] O. Lemon. Semantical foundations of spatial logic. In L. Aiello, J. Doyle, and S. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-96)*, pages 212–219. Morgan Kaufman, 1996.

[Loe76] M. H. Loeb. Embedding first order predicate logic in fragment of intuitionistic logic. *Journal of Symbolic Logic*, 41:705–718, 1976.

[Pau94] L.C. Paulson. *Isabelle: a generic theorem prover*, volume 828. Springer, 1994.

[PS98] I. Pratt and D. Schoop. A complete axiom system for polygonal mereotopology of the real plane. *Journal of Philosophical Logic*, 27:621–658, 1998.

[RCC92] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, 1992.

[She99] V. Shehtman. 'Everywhere' and 'here'. *Journal of Applied Non-Classical Logics*, 9(2-3):369–279, 1999.

[Ste00] J. G. Stell. Boolean connection algebras: a new approach to the region-connection calculus. *Artificial Intelligence*, (122):111–136, 2000.

[Tar56] A. Tarski. Sentential calculus and topology. In *Logic, Semantics, Metamathematics.* Oxford University Press, 1956.

[TB02] P. Torrini and B. Bennett. A spatial logic capable of representing connectedness and planarity. Submitted for publication, 2002.

[TS02] P. Torrini and J.G. Stell. Spatial concepts in an intermediate logic with propositional quantification. Submitted for publication, 2002.