AISB Convention 1999

Sixth
Automated Reasoning Workshop:
Bridging the Gap
between Theory and Practice

8-9 April 1999

University of Edinburgh, Edinburgh, Scotland

Organisers:	Rob Arthan, Lemma 1	rda@lemma-one.com
	Alan Bundy, Edinburgh	A.Bundy@ed.ac.uk
	Anthony Cohn, Leeds	agc@scs.leeds.ac.uk
	Michael Fisher, Manchester	M.Fisher@doc.mmu.ac.uk
	Alan Frisch, York (Chair of Org. Committee)	frisch@cs.york.ac.uk
	Ian Gent, Strathclyde	ipg@cs.strath.ac.uk
	A. Ireland, Heriot-Watt	A.Ireland@hw.ac.uk
	M. Kerber, Birmingham (Programme Chair, Editor)	M.Kerber@cs.bham.ac.uk
	H. J. Ohlbach, Kings College	ohlbach@dcs.kcl.ac.uk
	Toby Walsh, Strathclyde	tw@cs.strath.ac.uk

RELEVANT WEB SITES:

Workshop:	http://www.cs.bham.ac.uk/~mmk/AR99/
Workshop Series:	http://www.cs.york.ac.uk/~frisch/AutoReason/
AISB-Convention:	http://www.dai.ed.ac.uk/~geraint/aisb99/

Contents

Foreword

The 1999 **Workshop on Automated Reasoning – Bridging the Gap between Theory and Practice** is the sixth in a series with the aim to provide an informal forum for the automated reasoning community. It should bring together researchers from all areas of automated reasoning in order to foster links and facilitate cross-fertilisation of ideas among researchers from various disciplines; among researchers from academia, industry and government; and between theoreticians and practitioners.

In preparation of the workshop we have invited the submission of abstracts from the full breadth and diversity of automated reasoning, including topics such as logic and functional programming; equational reasoning; deductive databases; unification and constraint solving; the application of formal methods to specifying, deriving, transforming and verifying computer systems and in particular safety-critical systems; deductive and non-deductive reasoning, including abduction, induction, nonmonotonic reasoning, and analogical reasoning; commonsense reasoning; and the wide range of topics that fall under the heading of knowledge representation and reasoning.

The workshop programme consists of invited talks by Simon Colton, Maarten de Rijke, and Andrei Voronkov, of two panel sessions, one on "Diagrammatic Reasoning" with Peter Cheng, Mateja Jamnik, and Keith Stenning, another organised by Michael Fisher on "Proof in Combinations of Modal Logics" with Brandon Bennett, Clare Dixon, Enrico Franconi/Ian Horrocks, Ullrich Hustadt, and Maarten de Rijke. Furthermore we have joined events with the whole AISB convention and with the "Symposium on AI and Scientific Creativity". In order to offer optimal possibilities for an interactive workshop all participants can present their work in form of posters as well as in an open discussion forum.

We would like to thank AISB and in particular Geraint Wiggins not only for all their work in organising the AISB convention, but also for the flexibility and extra work they did in order to make special arrangements for our workshop to meet our needs best. We would like to thank the Compulog Network of Excellence for the financial support. Finally we would like to thank all the authors and in particular the invited speakers for their contributions.

The main objective of the abstracts collected here is to spread information about recent work in our community. We deliberately asked for short abstracts in order to make it feasible to read all of them in relatively short time. We hope that many will do so and will feel tempted to look up the full papers, which are referenced in the abstracts.

The Organisers

Part I

Invited Talks

Automated Discovery in Pure Mathematics

Simon Colton, Alan Bundy and Toby Walsh,
Institute of Representation and Reasoning
Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh. EH1 1HN. Scotland.
`simonco,bundy,tw@dai.ed.ac.uk`

The HR project aims to automate two important discovery processes which occur in mathematics before theorem proving happens, namely the making of the conjecture to be proved and the invention of the definitions in the conjecture statement. Our approach is to:

(1) Represent pure mathematics concepts as data-tables, eg. represent division of integers as a data-table with 3 columns, $[n, a, b]$, where $a, b, n \in \mathbf{N}$ and $a \times b = n$. Definitions for concepts can be generated when needed, using information about how they were constructed.

(2) Use production rules (PRs) to turn old concepts into new ones. HR has 10 production rules and uses parameters to detail exactly what to do. In figure 1, the **match** PR extracts rows where certain columns are the same (the parameters [1,2,2] say that col.1=col.1, col.2=col.2 and col.3=col.2). The **count** PR counts in how many rows the entries in column [1] appear. The **exists** PR removes columns, in this case, only keeping column [1]. Finally, the **split** PR, with parameters [c,=,v], extracts rows where column c has value v. The PRs are general enough to work in number theory, and to also find concepts such as complete graphs in graph theory and Abelian groups in group theory. Note that HR has been developed using group theory as the test domain.

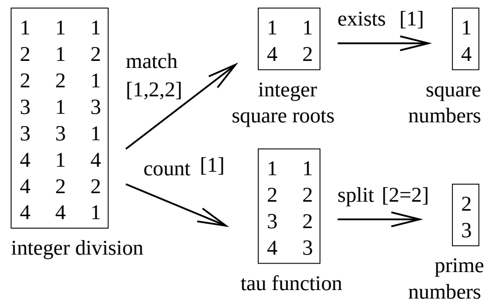


figure 1: construction of squares & primes

(3) Derive measures which estimate how interesting a concept is, and use the heuristic of basing new concepts only on the most interesting old ones. A concept's data-table can be used to describe the models HR is working with, eg. the output from the tau function in figure 1 describes integer 1 as 1, 2 as 2, 3 as 2 and integer 4 as 3. The *parsimony* measure of a concept is inversely proportional to the size of its data table, ie. more parsimonious descriptions are advantageous. These descriptions can be used to categorise HR's models by putting two models in the same category if described the same, eg. the tau function gives integers 1 to 4 this categorisation: [1],[2,3],[4]. The user can supply a 'gold standard' categorisation which they would like the concepts to achieve, eg. into squares and non-squares: [1,4],[2,3]. HR can then use the *invariance/discrimination* measures, which find the proportion of pairs of models categorised together/apart in the gold standard, which are categorised correctly by the concept. Given the isomorphic classification of groups up to order 6, HR found this function which categorises them correctly: $f(G) = |\{(a, b, c) \in G^3 : a * b = c \wedge b * c = a\}|$. A fourth measure is the *complexity* of a concept which counts the number of old concepts

appearing in its construction path, giving a rough idea of how long the definition will be, with more concise definitions being advantageous.

(4) Use empirical evidence to spot conjectures about concepts. The most common way HR does this is to notice that the data-table of a newly formed concept is exactly the same as an old concept. HR then makes the conjecture that the definition of the new concept is equivalent to the definition of the old concept, eg. $\forall$ groups, G, $predicate_1(G) \iff predicate_2(G)$.

(5) Enable HR to use theorem prover OTTER, [McC90], to prove some conjectures it makes. OTTER performs poorly with inductively defined concepts, so only a subset of the conjectures can attempted. However, some of these are non-trivial, eg. in group theory, HR spotted that for any group, G, and $\forall a, b, c \in G, (a{*}b = c \land a{*}c = b \land b = id \iff a{*}b = c \land c{*}a = b \land a{*}a = b)$. Instead of passing this verbatim to OTTER, HR first tries to extract and prove any prime implicates (conjectures where no subset of the premises imply the goal). For example HR extracts this: $a * c = b \land b = id \rightarrow c * a = b$, which OTTER proves. HR can use the prime implicates to prove the overall conjecture, and will use them to prove other conjectures later. If a conjecture cannot be proved, HR passes it to model generator MACE, [McC94], which is asked to find a counterexample of size 1, then 2, etc. If a counterexample is found, it is added to the theory, and all future conjectures will be based on the extra data.

(6) Complete a cycle of mathematical activity by assessing the conjectures and using the average interestingness of the conjectures a concept appears in to estimate the interestingness of the concept itself. A conjecture could be interesting if OTTER found it difficult to prove - HR uses OTTER's proof length statistic to determine this. Also, a conjecture could be interesting if it is surprising, ie. the (supposedly) equivalent concepts look quite different - HR calculates the proportion of concepts appearing in one, but not both, construction paths.

HR can bootstrap theory formation, ie. starting with just the axioms of a finite algebra like group theory, it uses MACE to find initial models of the algebra, invents definitions, spots conjectures and uses OTTER to prove them and MACE to disprove them, which introduces counterexamples. HR's biggest success so far was in number theory, [Col99], where it invented the concept of refactorable numbers: $1, 2, 8, 9, 12, 18, 24, 36, 40, 56, 60, 72, 80, 84, 88, 96, \ldots$ These have the appealing definition of being those integers where the number of divisors is itself a divisor. This concept was novel because it was missing from, and subsequently added to, the online-encyclopedia of integer sequences, [Slo]. HR also spotted some conjectures about refactorables, eg. they are congruent to 0, 1, 2 or 4 mod 8, which we have subsequently proved, along with some results of our own, eg. all odd refactorables are squares. For further details about HR, please see [CB] or the web page: `http://dream.dai.ed.ac.uk/group/simonco/hr`.
This work is supported by EPSRC grant GR/L 11724 and EPSRC studentship GR/K/65706.

References

[CB] S Colton and A Bundy. HR: Automatic concept formation in pure mathematics. In *IJCAI '99*.

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

[McC90] W McCune. The OTTER user's guide. Technical Report ANL/90/9, ANL, 1990.

[McC94] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, ANL, 1994.

[Slo] N Sloane. *Online Encyclopedia of Integer Sequences*. www.research.att.com/~njas/sequences.

Combining Reasoning Systems

Maarten de Rijke
ILLC, University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands
e-mail: `mdr@wins.uva.nl`
WWW: `http://www.illc.uva.nl/~mdr/`

Why Combine Logics?

Combining logics for modeling purposes has become a rapidly expanding enterprise that is inspired mainly by concerns about modularity and the wish to join together different kinds of information. As any interesting real world system is a complex composite entity, decomposing its descriptive requirements (for design, verification, or maintenance purposes) into simpler, more restricted reasoning tasks is appealing as it is often the only plausible way forward. It would be an exaggeration to claim that we have a thorough understanding of 'combined methods.' Nevertheless, a core body of notions, questions and results has emerged for an important class of combined logics, and we are beginning to understand how this core theory behaves when we try to apply it outside this particular class.

Does the idea of combining logics actually offer anything new? Some of the possible objections can be justified. Logical combination is a relatively new idea: it has been not yet systematically explored, and there is no established body of the results or techniques. Nonetheless, there is a growing body of logic-oriented work in the field, and there are explorations of their uses in AI, computational linguistics, automated deduction, and computer science. An overly critical reaction seems misguided.

The plan of this abstract is as follows. We start with a discussion of a class of problems typically considered in the are. We then take a brief look at actual implementations of combined logics.

Transfer Problems

Let $\mathbf{L}_1$ and $\mathbf{L}_2$ be two logics—typically, these are special purpose logics with limited expressive power, as it often does not make sense to put together logics with universal expressive power. Let P be a property that logics may have, say decidability, or axiomatic completeness. The *transfer problem* is this: if $\mathbf{L}_1$ and $\mathbf{L}_2$ enjoy the property P, does their combination $\mathbf{L}_1 \oplus \mathbf{L}_1$ have P as well? Transfer problems belong to the main mathematical questions that logicians have been concerned with in the area of combining logics.

When, and for which properties do we have transfer or failure of transfer? As a rule of thumb, in the absence of interaction between the component logics, we do have transfer; here, absence of interaction means that the component languages do not share any symbols, except maybe the booleans and atomic symbols. Properties that do transfer in this restricted case include the finite model property, decidability, and (under suitable restrictions on the classes of models and the complexity class) of complexity upper bounds.

The positive proofs in the area are usually based on two key intuitions: *divide and conquer* and *hide and unpack*. That is: try to split problems and delegate sub-problems to the component logics; and when working inside one of the component logics view information relating to other component logics as alien information and 'hide' it—don't unpack the hidden information until we have reduced a given problem to a sub-problem in the relevant component logic. Neither of these key intuitions continue to work in the presence of interaction. For instance, consider two modal languages $\mathcal{L}_1$ and $\mathcal{L}_2$ with modal operators $\Box$ and $\blacksquare$, respectively; there are logics $\mathbf{L}_1$ and $\mathbf{L}_2$ in $\mathcal{L}_1$ and $\mathcal{L}_2$ whose satisfiability problem is in NP, while the satisfiability problem for the combined language *plus* the interaction principle $\Box p \to \blacksquare p$ is undecidable.

Combinations at Work

Does combining logics work for actual reasoning systems? That is: can existing tools for component logic be put together to get tools for combined logics? Obviously, the *re-use* of tools and procedures is one of the key motivations underlying the field. Now, one cannot put together any proof procedures for two logics in a uniform way. First, 'proving' can have different meanings in different logics: (semi-)deciding satisfiability or validity, computing an instantiation, or generating a model. Second, it is not clear where to "plug in" the proof procedure for a logic $\mathbf{L}_1$ into that for a second logic $\mathbf{L}_2$; a proof procedure may have different notions of valuations, or of proof goals.

So what can one do? One way out is to impose special conditions on the calculi that one wants to combine [BG98]. Another possibility, in the case of modal logics, is to use a translation-based approach to theorem proving, by mapping all component logics into a common background logic. [GS96] provide an interesting example by combining efficient propositional decision procedures into a decision procedure for the modal logic $\mathbf{K}$.

There are quite a few successful particular instances of combined logics, where we have no problems whatsoever in putting together tools; see [AAdR99, KdR99]. By and large, however, we don't have a good understanding of how to proceed. Further experimentations are needed, both locally, and network based, so that at some stage we will be able to plug together tools without having to be the designer or engineer of the systems.

References

[AAdR99] M. Aiello, C. Areces, and M. de Rijke. Spatial reasoning and image retrieval. Submitted.

[BG98] B. Beckert and D. Gabbay. Fibring semantic tableaux. In *Proc. Tableaux'98*, pp. 77–92, 1998.

[Gab99] D.M. Gabbay. *Fibring Logics*. Oxford University Press, 1999.

[GS96] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—The case study of modal K. In *Proc. CADE-13*, pp. 583–597, 1996.

[KdR99] N. Kurtonina and M. de Rijke. Hybrid logics and linguistic inference. Submitted.

[KW97] M. Kracht and F. Wolter. Simulation and transfer results in modal logic. *Studia Logica*, 59:149–177, 1997.

Herbrand's Theorem, Automated Reasoning and Semantic Tableaux

Andrei Voronkov
Computer Science Department,
University of Manchester,
Oxford Road,
Manchester, M13 9PL, England
e-mail: `voronkov@cs.man.ac.uk`
WWW: `http://www.cs.man.ac.uk/fmethods/people/voronkov.html`

Numerous results related to the Herbrand theorem and the foundations of semantics tableaux were obtained recently. These results shed a new light on the nature of such methods and computational complexity of several problems related to the Herbrand theorem. The aim of our talk is to analyze these results, sketch new research directions and pose some open problems.

The main questions we ask in the talk are the following.

- Are tableaux and related methods inherently inefficient?
- In what cases are these methods efficient?
- In what cases and how can we increase efficiency of these methods?

This talk is structured as follows.

1. We give a short description of the tableau and related method and consider a typical proof-search procedure for such methods. We analyze this procedure and point out two problems, instantiation of a quantifier-free formula to a valid formula, and strategies for selecting multiplicity.
2. We explain several results on the complexity of the formula instantiation problem and further discuss some results about the strategies for multiplicity.
3. We mention some related open problems and new interesting research directions.

Many people contributed to these recent results, including Anatoli Degtyarev, Margus Veanes, Yuri Matiyasevich, Yuri Gurevich, David Plaisted, Paliath Narendran, Paul Voda and Jan Komara. Some key references are listed below. Most results and references can be found in [Vor98a, Vor98b, Vor98c]. A book by Margus Veanes and the author is under preparation.

References

[DGN+98] A. Degtyarev, Yu. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. The decidability of simultaneous rigid E-unification with one variable. In T. Nipkow, editor, *Rewriting Techniques and Applications, RTA'98*, volume 1379 of *Lecture Notes in Computer Science*, pages 181–195. Springer Verlag, 1998.

[DMV96] A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification and related algorithmic problems. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 494–502, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[DV96a] A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 503–512, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[DV96b] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

[DV98] A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. *Journal of Automated Reasoning*, 20(1):47–80, 1998.

[GV97] Yu. Gurevich and A. Voronkov. Monadic simultaneous rigid E-unification and related problems. In P. Degano, R. Corrieri, and A. Marchetti-Spaccamella, editors, *Automata, Languages and Programming. 24th International Colloquium, ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 154–165, Bologna, Italy, July 1997.

[Vea97a] M. Veanes. *On Simultaneous Rigid E-Unification*. PhD thesis, Uppsala University, 1997.

[Vea97b] M. Veanes. The undecidability of simultaneous rigid E-unification with two variables. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational Logic and Proof Theory. 5th Kurt Gödel Colloquium, KGC'97*, volume 1289 of *Lecture Notes in Computer Science*, pages 305–318, Vienna, Austria, 1997.

[Vea98] M. Veanes. The relation between second-order unification and simultaneous rigid E-unification. In *Proc. Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 264–275, Indianapolis, Indiana, January 1998. IEEE Computer Society Press.

[VK95] P.J. Voda and J. Komara. On Herbrand skeletons. Technical report, Institute of Informatics, Comenius University Bratislava, July 1995.

[Vor97] A. Voronkov. Strategies in rigid-variable methods. In M.E. Pollack, editor, *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, volume 1, pages 114–119, Nagoya, Japan, August 23–29 1997.

[Vor98a] A. Voronkov. Herbrand's theorem, automated reasoning and semantic tableaux. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 252–263. IEEE Computer Society Press, 1998.

[Vor98b] A. Voronkov. Herbrand's theorem, automated reasoning and semantic tableaux. UPMAIL Technical Report 151, Uppsala University, Computing Science Department, February 1998.

[Vor98c] A. Voronkov. Simultaneous rigid E-unification and other decision problems related to Herbrand's theorem. UPMAIL Technical Report 152, Uppsala University, Computing Science Department, February 1998.

[Vor99] A. Voronkov. The ground-negative fragment of first-order logic is Π_2^p-complete. *Journal of Symbolic Logic*, 1999. To appear.

Part II

Contributions

Spatio-Temporal Reasoning with Multi-Dimensional, Multi-Modal Logics and Constraints

Brandon Bennett
Division of Artificial Intelligence
School of Computer Studies
University of Leeds, Leeds LS2 9JT, England
`brandon@scs.leeds.ac.uk`

In recent work I have been exploring possibilities of using multi-dimensional, multi-modal logics as a basis for spatio-temporal knowledge representation and inference. In [Ben96] I showed how the logic $S4$ could be used to encode topological information in such a way that a tractable decision procedure could be obtained for reasoning with a significant class of topological relations. I am now trying to extend this approach to cover temporal as well as spatial information.

A '2-dimensional' extension of the modal logic $S5$ was given in [Seg73]. This contains two modal operators, $\boxminus$ and $\boxbar$, each of which obeys all the usual axioms of the modal logic $S5$. These axioms are supplemented by a further schema which ensures that the two modalities commute: $\boxminus \boxbar \phi \leftrightarrow \boxbar \boxminus \phi$. An attractive idea is that a bi-modal logic can represent spatio-temporal information by employing one modality as a spatial modality and the other as a temporal modality. Thus, we employ $\boxminus$ to range over spatial points and $\boxbar$ to range over time indices and we use the 'propositional' constants of our language to denote spatio-temporal objects. Under this interpretation a formula $\boxminus(r \rightarrow s)$ means that (at the current time) object r is a part of object s. Within the two-dimensional logic it is straightforward to put space and time together and represent, e.g. "During breakfast-time the newspaper was in the kitchen" by the formula $\boxbar\ (\beta \rightarrow \boxminus\ (\mathbf{n} \rightarrow k))$. This gives us a quantifier-free spatio-temporal representation.

Using this encoding we can employ a modal theorem prover to verify inferences such as:

```
always(part(leeds,yorkshire)),
always(part(yorkshire,uk)),
future(overlap(sea,leeds))
```
$\models$ `future(overlap(sea,uk))`

In this example the `future` operator can be treated as a simple $S5$ possibility but in many cases of temporal reasoning one has to take account of both past and future and of the ordering of time points. Any of the many available tense logics could be used to characterise the temporal dimension. However, problems of deductive completeness and computational complexity may arise due to the interaction of the temporal logic with the spatial dimension. One solution might be to employ a calculus similar to that of [DFW98] which was devised for reasoning with temporal logics of knowledge; but so-far I have only employed the much simpler technique of *layering*.

A *layered* multi-modal logic is one where restrictions are placed on the nesting order of the different modalities in the language. A typical layering restriction would be to require that temporal modalities lie outside the scope of spatial modalities. This would allow one to represent tensed spatial constraints; but not specify temporal constrains which vary over space. Whereas tensed spatial relations are very common in natural language, the latter kind of constraint is not so common. A simple layered logic is obtained by using two $S5$ modalities where all occurrences of one modality must lie outside all occurrences of the other. In this case there is no interaction between the modalities and models for the inner modality can be 'laid side by side' to form a model for the bi-modal system.

A complementary technique which I am exploring is to combine tableau-style modal theorem proving with a modular constraint satisfaction procedure which handles certain restrictions on the possible world structure. Because modal logic is being used to represent spatio-temporal information modal indices can be regarded as coordinates and constraints on admissible model structures typically take the form of geometrical constraints. In [Ben96] I encoded equational and disequational constraints on a set of possible worlds (representing spatial points) by means of an $S5$ operator. A formula $\Box\phi$ says that the constraint ϕ is satisfied at all points and $\Diamond\phi$ says that there is some point satisfying ϕ. This can be generalised to conditions of the form $\Box(\phi \rightarrow (\Diamond\psi_1 \wedge \ldots \wedge \psi_n))$ meaning that every 'ϕ point' is associated with certain 'ψ points'. In such a case additional geometrical conditions holding among the ϕ point and the ψ_i points could be handled by a separate constraint handling procedure adjoined to a tableau theorem prover. This method enables us to give a neat encoding of the spatial constraint which holds between an arbitrary region and its *convex-hull*. Specifically, whenever a point p belongs to the convex-hull of some region r, there must exist two further points $x_1, x_2 \in r$ such that p lies on the line between x_1 and x_2. The relation of betweenness is easily represented as an equational constraint on the coordinates of the sample points.

It must be recognised that there are considerable difficulties facing my approach to spatio-temporal reasoning. Specifically, it is currently a matter of argument whether multi-modal can really lead us to practical decision procedures. For instance [Kra95] has shown that many seemingly simple multi-modal logics are undecidable. On the other hand, the expressive power of these formalisms means that much information can be conveyed by formulae having a very shallow structure. This suggests that it may be possible to find fragments of multi-modal languages which are both tractable and useful.

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. WWW address `ftp://ftp.mpi-sb.mpg.de/pub/igpl/Journal/V4-1/index.html`.

[DFW98] C Dixon, M Fisher, and M Wooldridge. Resolution for temporal logics of knowledge. *Journal of Logic and Computation*, 8(3), 1998.

[Kra95] M. Kracht. Highway to the danger zone. *Journal of Logic and Computation*, 5(1):93–109, 1995.

[Seg73] K. Segerberg. Two-dimensional modal logic. *Journal of Philosophical Logic*, 2:77–96, 1973.

Towards Fine-Grained Proof Planning with Critical Agents

Christoph Benzmüller and Volker Sorge
Universität des Saarlandes {chris|sorge}@ags.uni-sb.de

Agent Based Command Suggestion Mechanism In order to support users within large proofs, most interactive theorem provers have some limited capability to suggest parameters of choosen commands or even commands themselves. However, those mechanisms generally have the shortcomings that they use inflexible sequential computation strategies (i.e., although most parameters influence the suggestions of each other they are suggested in some order such that parameters choosen earlier influence those required later, but not vice versa) and do not sufficiently use available system resources by computing suggestions inbetween single interaction steps.

We have developed a mechanism that suggests commands, applicable in the current proof state together with a suitable argument instantiations [BeSo98]. It is based on two layers of societies of autonomous, concurrent agents which steadily work in the background of a system and dynamically update their computational behavior to the state of the proof and/or specific user queries to the suggestion mechanism (cf. Fig. 1). By exchanging relevant results via blackboards the agents cooperatively accumulate useful command suggestions which can be heuristically sorted and presented to the user.

The architecture has been prototypically implemented in the ΩMEGA mathematical assistant system [Ome97] for interactive tactical theorem proving in natural deduction calculus. In [BeSo99] we have enriched the architecture with resource concepts that on the one hand enable the lower layer agents to adjust their computations with respect to their performance in the past and to the logical structure of the proof. On the other hand the behaviour of whole agent societies can be adapted according to the resources available to the overall system.

Although, the mechanism has been developed in the context of interactive theorem proving within ND-calculus it is neither restricted to a particular calculus nor to a solely interactive system. In fact, it can directly be used to compute possible applicable rules for some automated proving procedure or even be automated itself by introducing some limited automation and backtracking capabilities in the mechanism. In the remainder of this abstract we both hint at the latter and describe potential applications of the mechanims within automated proof planning.

Tackling Problems Automatically When automating the suggestion mechanism in a way that always the heuristically preferred suggestion is automatically executed and that backtracking over the single suggestions can be performed, some small proof problems can already be solved fully automatically. In fact, some problems, such as $(p_{o\to o}\ (a_o \wedge b_o)) \Rightarrow$

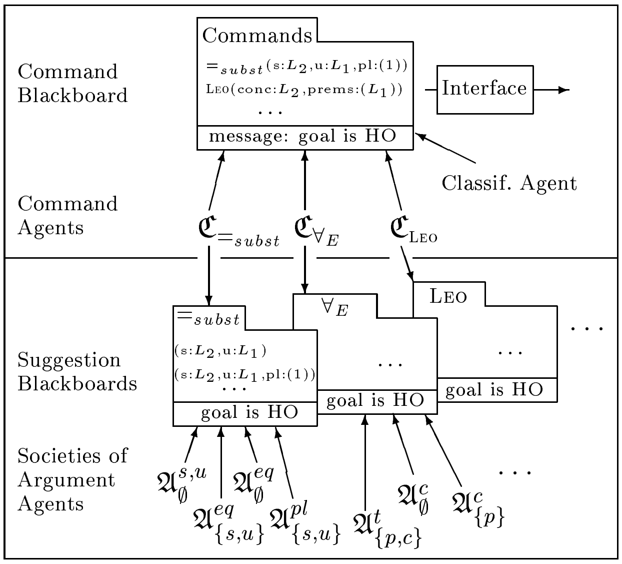


Figure 1: The two layered suggestion mechanism.

$(p\ (b \wedge a))$[1] can be solved even without the need for backtracking. Since the agent mechanism enables to work on several subproblems in parallel the approach can be viewed as a weak form of concurrent, tactic based theorem proving. Surely, tactic based theorem proving based on an underlying natural deduction approach stretches huge search spaces when considering non-trivial examples, such that the problem solving capabilities of such a low level approach will certainly be limited. But note that we can flexibly alter and exchange the rules and tactics underlying our approach in order to influence the size of the search space. It especially appears to be interesting to apply our proposed approach to exactly the rules provided by the Normal Form Natural Calculus [SiBy98] which aims at a fruitful restriction of the search space by allowing only for the construction of ND-proofs that have a particular normal form. The additonal side conditions of the rules in this approach, which restrict and guide their applicabilty, can be easily specified within our approach.

Suggesting Methods in Proof Planning The approach is not restricted to specify applicability conditions for rules and tactics only, but can analogously be employed for proof methods as used in proof planning approaches (e.g., see [Bun88]). So far the specification of the argument agents for rules and tactics describes structural properties of single arguments as well as structural dependencies between the different arguments. Similarly we can specify agents which check the particular pre-conditions of a proof method, i.e., check for proof lines matching with those required for a method to be applicable or verify additional application conditions of the method. Thus the hope is that we can at least to some extend exploit concurrency even within a traditional proof planner when computing the applicable methods with respect to the given proof goal by cooperating argument agents in each proof step.

To guide the proof planners selection of methods heuristical knowledge can be realised with the suggestion mechanism in different ways: (i) Resource allocations: By adjusting the resource allocations for the agents belonging to the proof methods one influences the suggested

[1]The example is small but quite challenging as it involves the application of extensionality principles.

steps in the proof planning approach in the same way as adjustments influence the suggested commands in the current approach. Within the resource adaptive extension [BeSo99] the proof methods can even adapt their own resource allocations with respect to the current goal and their experience in the past. (ii) Sorting criteria: Sorting criteria are employed to select particular suggestions from the suggestion blackboards. E.g., among all suggestions on the command blackboard we prefer those that might yield the largest proof steps. Any modification to these sorting criterias will obviously influence the proof planners selection of methods in each planning step. (iii) Informed activation & deactivation: In our approach agents are aware of the mathematical theory the command they work for is associated with. E.g., agents that work for a command which applies a tactic that deals with computations on natural numbers will automatically retire if the proof is in a theory where the concept of numbers is not even defined. By classifying proof methods with respect to different mathematical theories on can restrict the computations of the proof planner when considering planning goals belonging to a particular theory.

It seems to be interesting to both implement and experiment with this architecture and furthermore check how far the approach can be coupled with traditional proof planning approaches.

References

[BeSo98] Christoph Benzmüller and Volker Sorge. A Blackboard Architecture for Guiding Interactive Proofs. In F. Giunchiglia, editor, *Proceedings of AIMSA'98*, LNAI 1480, pages 102–114. Springer, Berlin, Germany, 1998.

[BeSo99] Christoph Benzmüller and Volker Sorge. Critical Agents Supporting Interactive Theorem Proving. Seki Report SR-99-02, Computer Science Department, Universität des Saarlandes, 1999.

[Bun88] Alan Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In E. Lusk and R. Overbeek, editors, *Proceedings of CADE–9*, *LNCS* 310, Argonne, USA, 1988. Springer, Berlin, Germany.

[Ome97] The ΩMEGA Group. ΩMega: Towards a Mathematical Assistant. In W. McCune, editor, *Proceedings of CADE–14*, *LNAI* 1249. Springer, Berlin, Germany, 1997.

[SiBy98] Wilfried Sieg and John Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica*, 60(1):67–106, January1998.

Clausal Resolution for Branching-Time Temporal Logics and Temporal μ-Calculus.

Alexander Bolotov
Department of Computing and Mathematics
Manchester Metropolitan University
Manchester M1 5GD, UK.
Email: `A.Bolotov@doc.mmu.ac.uk`
WWW : `www.doc.mmu.ac.uk/STAFF/A.Bolotov`

I will provide an outline of two topics I have been recently concentrated on. They extend the clausal resolution approach to linear-time temporal logic [Fis91, Fis96, Dix96] to more expressive frameworks of branching-time [Em90] and temporal μ-calculus [St89].

- The clausal resolution method was extended to CTL [BF99] and has been recently proposed for the full branching-time logic CTL* [BDF99] where two major components, namely, a concept of a normal form (together with a procedure to transform CTL* formulae into their normal forms) and a temporal resolution rule, have been introduced and justified.
- Based on some features of the language of a normal form used for linear-time temporal logic (and on the fact that it is strictly more expressive than the language of this logic itself [Fis96]), I am currently incorporating this language to capture a normal form for the linear-time temporal μ-calculus. This will serve as a basis for the developing of a clausal resolution procedure for the linear-time temporal μ-calculus.

References

[BF99] A. Bolotov and M. Fisher. A Clausal Resolution Method for CTL Branching-Time Temporal Logic. To appear in *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 11, Tailor & Francis, 1999, pages 77-93.

[BDF99] A. Bolotov and M. Fisher. Clausal Resolution for CTL* Branching-Time Temporal Logic. Submitted.

[Dix96] C. Dixon. Search Strategies for Resolution in Temporal Logics. In *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE)*, Lecture Notes in Computer Science. Springer-Verlag: Heidelberg, Germany, August 1996.

[Em90] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.

[Fis91] M. Fisher. A Resolution Method for Temporal Logic. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, 1991. Morgan Kaufman.

[Fis96] M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *Journal of Logic and Computation*, 7(4), August 1997.

[St89] C. Stirling. Modal and Temporal Logics. In *Handbook of Logic in Computer Science.* Oxford University Press, 1992.

Reasoning about Computer Languages by Automated Mathematical Induction

Richard J. Boulton
Division of Informatics, The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland
e-mail: rjb@dai.ed.ac.uk
WWW: http://www.dai.ed.ac.uk/~rjb/

Introduction

My research is directed towards automating formal verification, especially reasoning about programming languages and hardware description languages. My current focus is on automating mathematical induction in the HOL interactive theorem prover by interfacing it to the CLAM proof planner. My intention is to combine this research with earlier work on defining the syntax and semantics of computer languages in HOL to produce a tool in which semantically based transformations on programs and hardware descriptions can be verified semi-automatically.

Syntax and Semantics of Computer Languages

The abstract syntax of programming languages (and other formal languages) can be represented as recursive data types, with one type for each syntactic category. For example, a syntactic category of simple arithmetic expressions might be represented by the following type (expressed in the syntax of the Standard ML programming language):

```
datatype exp = num of int | var of string | plus of exp * exp
             | minus of exp * exp | times of exp * exp
```

Thus an arithmetic expression is a numeric constant, a variable, or an arithmetic operator applied to two expressions. The type `exp` is recursive and has two base constructors `num` and `var`, and three step constructors `plus`, `minus`, and `times`.

In a similar way, the abstract syntax of a language can be formalised in a theorem prover by defining recursive types in the logic. Having done this, there are a number of ways of defining the semantics of the language. Probably the most popular way at present is to formalise the structural operational semantics of the language using inductively defined relations over the types that represent the abstract syntax. An alternative approach is to define recursive functions over the types, the results of which denote the meaning of the language constructs.

Proving Properties by Induction

When the operational semantics approach is taken, proofs of properties of the language are typically by rule induction. When recursive functions are used for the semantics, mathematical induction over the structure of the language is used. Since research in automating induction for recursive functions is much more mature than for rule induction, I am focusing on the use of recursive functions for language semantics.

I have developed a tool that takes a high-level specification of the semantics of a language and produces corresponding definitions in the HOL system [Bou98]. The original specification is sufficiently abstract that other theorem provers could be targeted instead. A crucial feature of the tool is its library of information about auxiliary functions used in the semantics. This information should help considerably in automating proofs about the semantics.

My main work at present is in interfacing the CLAM proof planner to the HOL system [BSBG98], not only for proving properties about languages, but to automate induction in general. Induction is a common feature in verification proofs, e.g. for recursion and loops in programs, and for reasoning about time and parameterised designs in digital circuits. The interfacing project has entered the testing phase and one example area I intend to pursue is using CLAM to verify transformations on code fragments relative to the semantics of the language in which the code is written.

Automating Induction for Mutual Recursion

Many languages have syntactic categories that are mutually dependent, e.g. Standard ML has declarations that involve expressions while expressions may themselves contain local declarations. In such situations the types used to represent the abstract syntax are mutually recursive and the semantic functions are mutually recursive. Mutually recursive functions present special difficulties for the automation of induction. In particular, the induction hypothesis may involve a different function to the induction conclusion. I have recently been working on this problem, motivated by the desire to automate the verification of transformations on languages that have mutually dependent syntactic categories.

References

[Bou98] R. J. Boulton. Generating embeddings from denotational descriptions. In Grundy and Newey [GN98], pages 67–86.

[BSBG98] R. Boulton, K. Slind, A. Bundy, and M. Gordon. An interface between Clam and HOL. In Grundy and Newey [GN98], pages 87–104.

[GN98] J. Grundy and M. Newey, editors. *Proceedings of the 11th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'98)*, volume 1479 of *Lecture Notes in Computer Science*, Canberra, Australia, September/October 1998. Springer.

Diagrammatic Representations for Efficient Reasoning and Effective Learning

Peter C-H. Cheng
ESRC Centre for Research in Development, Instruction and Training
Department of Psychology, University of Nottingham
University Park, Nottingham, NG7 2RD, UK
email: peter.cheng@nottingham.ac.uk
Tel: (UK.) +44 115 9515330

One of the major lessons of cognitive science is the fundamental role that representations have in cognition. Different representations for the same problem can make finding a solution more than an order of magnitude harder. Diagrams have been shown to confer benefits in problem solving compared with informationally equivalent propositional representations.

Two examples of work on the nature diagrammatic representations will be considered to illustrate some of the general issues concerning the role of representations in cognition. The first example focuses on a specific task to consider the effects of different representations in quite traditional information processing terms. The second example raises broader questions about the role of representations in reasoning and learning in real domains, which requires consideration of representations in broader ontological, structural and functional terms.

Discovering laws from data

Many models of the processes of scientific discovery have been proposed, including those which attempt to simulate the induction of laws from sets of data. For instance, the BACON program can find relations such as Kepler's law of planetary motion or Black's thermodynamic law, given numerical data [LaSi+87]. The system finds the laws by recursively looking for patterns in the data and proposing algebraic relations amongst the variables to match.

Such models of scientific discovery rely on the available formalisms in AI and have little to do with the representations really used in those episodes of discovery. Examination of the notebooks and published material of the original scientists shows that diagrams were essential in most of the discoveries. Kepler, Galileo and Newton all used geometric diagrams as the basis for their reasoning. Thus, to understand how their discoveries were really made it is essential to build models which use representations that are equivalent to those available to the original scientists.

Cheng and Simon [ChSi95] have done modelling of this kind in a system called HUYGENS. The key was to pick a method of representing internally, within the program, the information that was available in the diagrams of the chosen domains. The diagrams were geometric and the inferences were made by manipulating line segments with respect to each other. Thus, rather than storing data as discrete numeric values, for each item of data the system has a pair of values corresponding to the beginning and end points of a line segment for that item, on some arbitrary number line. This simple change meant that quite different regularity spotters, operators and heuristics had to be defined. Runs of HUYGENS on data for different domains, showed that the overall size of the search space was substantially less than for a system using a conventional algebraic representation. By changing the basic structure of the representation, the difficulty of the task of law induction was reduced.

A general conclusion that can be drawn for from this and related work, is that the choice of formalism for studying particular cognitive or computational phenomena will be critical. Careful consideration should be given to whether or not the available representations are appropriate and/or realistic. The problems facing techniques for reasoning in various domains may be due as much to the choice of representation itself, as to some inherent characteristics of the phenomena.

Reasoning and learning in substantial complex domains

Considering domains such as basic electricity, probability theory, particle collisions and Galilean kinematics brings into starker relief the demands placed on the representational systems to support effective reasoning and learning. In such domains information comes in many different forms, has multiple interacting levels, and can be viewed from alternative perspectives. For instance, in electricity there are circuit diagrams to show the topology of networks and algebraic equations to encode the laws governing the behaviour of circuits. There are different levels of abstraction, ranging from the general laws of electricity through to specific descriptions of the operation of particular configurations of networks. One may consider how circuits work in terms of formal algebraic relations amongst variables or one can reason about electricity in causal terms.

To study the role of representations in such complex domains it is necessary to compare different representational systems across a wide range of problem solving tasks and to consider how effectively they support learning. For some domains, alternative representations can be found it the history of science, where the systems used by the original scientists have been subsequently superseded by other representations. Examples include the geometric diagrams used in early physics, as modelled in the HUYGENS system discussed above, which have now been replaced by the development of algebra. Giving learners these diagrams rather than algebraic equations improves their learning [Che96b]. Unfortunately, it is often case that just a single conventional representation exists for a domain. For example, algebra (along with some supplementary diagrams) dominates probability theory. Some instructors have attempted to use geometric approaches, but those representations are only applicable to a limited range of problems. However, this state of affairs presents us with an opportunity to design new representational systems, based on theories about representations. The successful design of a representation that effectively supports reasoning and learning argues for the acceptability of the principles used in the design. This may be a profitable way to study the role of representations in cognition.

Under this general methodology new systems representations have been designed for electricity and probably theory to compare with the traditional algebraic approaches to these domains. The representations are diagrammatic and are members of a general class called Law Encoding Diagrams, LEDs, to which the geometric diagrams of the early physicists also belong [Che96a]. LEDs use diagrammatic constraints to encode the laws of a domain in a specific way, such that each instantiation (drawing) of a LED represents just a single phenomenon and one case of the laws. Thus they make available information at concrete and abstract levels simultaneously. Being diagrammatic they can be interpreted in different ways that are consistent with different perspectives on a domain. Empirical evaluations of problem solving and learning with the new LED representation for electricity have demonstrated that it is effective [Che99a]. General characteristics of good representations for substantial domains were derived from these studies and were subsequently used to design a diagrammatic representation for probability theory [Che99b].

The general implication of this work is that what constitutes an effective representation cannot just be considered in terms of the usual concepts of complex information processing (operators, heuristics, methods of indexing information). Rather, the overall ontological, structural and functional properties of the representations must be addressed, in order to fully characterize the role that they play in cognition.

References

[Che96a] Cheng, P. C.-H. (1996a). Scientific discovery with law encoding diagrams. *Creativity Research Journal*, 9(2&3), 145-162.

[Che96b] Cheng, P. C.-H. (1996b). Law encoding diagrams for instructional systems. *Journal of Artificial Intelligence in Education*, 7(1), 33-74.

[Che99a] Cheng, P. C.-H. Electrifying representations for learning: An evaluation of AVOW diagrams for electricity. *submitted.*

[Che99b] Cheng, P. C.-H. Representational analysis and design: What makes an effective representation for learning probability theory? *submitted*

[ChSi95] Cheng, P. C.-H., & Simon, H. A. (1995). Scientific Discovery and Creative Reasoning with Diagrams. In S. Smith, T. Ward, & R. Finke (Eds.), *The Creative Cognition Approach* (pp. 205-228). Cambridge, MA: MIT Press.

[LaSi+87] Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific Discovery: Computation Explorations of the Creative Processes.* Cambridge, MA: MIT Press.

Prosper: Proof and Specification Assisted Design Environments

Louise Dennis
Research Associate, Prosper project
Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ
email: louised@dcs.gla.ac.uk
WWW: http://www.dcs.gla.ac.uk/prosper

Existing proof tools need experts in logic and are not integrated into established design flows. Prosper is an ESPRIT funded project[1] spanning the universities of Glasgow, Edinburgh, Cambridge and Karlsruhe as well as the companies IFAD and Prover. It is coordinated from Glasgow by Prof. Tom Melham. It aims to develop the technology needed to overcome the barrier imposed by the learning–curve of current tools. We have the concrete objective of producing two prototype design tools, one for hardware and one for software, each providing user–friendly access to formal techniques.

Powerful formal techniques are already available in current mechanised proof tools, but they are not accessible to working designers. Prosper aims to bring the benefits of mechanised formal specification and verification to industrial system designers.

An essential part of the project is that the prototype design tools it produces are not "one–off" efforts but that they are constructed on top of an open proof architecture developed on the project. This architecture with efficient plug–in proof tools produces a general long–lasting solution that gains in value as more effective proof plug–ins are integrated and as other CAD/CASE tools are linked in. There are formal analysis techniques that may already be used effectively in industry, if well integrated with conventional design tools. New proof plug–ins will be developed in future that broaden the range of application areas that can benefit from formal analysis, and the open proof architecture will provide a fast–track means of making these tools available to designers.

The architecture proposed centres around the idea of a *core proof engine* which supports a well–defined API to developers of CAD/CASE tools and a well-defined interface to designers of other proof/verification tools that can be used as plug–ins.

In particular the project proposes to integrate Prosper technology into the IFAD VDM–SL toolbox using the Prover proof tool as a plug–in. It is hoped that in this way many of the proof obligations generated by the toolbox can be automatically discharged. We also propose to build a hardware verification workbench that will support both the Verilog and VHDL hardware description languages and will give hardware designers access to a combination of theorem–proving and model–checking tools. At the same time we are investigating the

[1]project number: LTR 26241

translation of hardware properties expressed in natural language into CTL.

At present a small prototype core proof engine has been built based upon the HOL theorem prover. This has been augmented with an abstract data-type which allows communication with both potential plug-ins and CAD/CASE systems. A simple Java front end has been created as an example of how a CAD/CASE system might interact with the system. Several prototype plug–ins have also been created from a BDD package, the SMV model checker and the SVC decision procedure. The project work in Glasgow over the next few months will concentrate on improving this prototype. Other partners will be focusing on integrating the architecture into industrial scale CAD/CASE tools.

Proof Methods for Multi-Agent Systems

Clare Dixon

Logic and Computation Group,
Department of Computing and Mathematics
Manchester Metropolitan University, Manchester M1 5GD, U.K.

Email: C.Dixon@doc.mmu.ac.uk
WWW [personal]: www.doc.mmu.ac.uk/STAFF/C.Dixon
WWW [group]: www.doc.mmu.ac.uk/RESEARCH/LoCo

Multi-agent systems are a growing area of interest in both industry and academia [WJ95]. Such systems are complex, often characterised by being dynamic, distributed, or reactive where qualities such as beliefs, desires, intentions or knowledge of agents may need to be expressed. To specify such systems, complex frameworks have been proposed, see for example [HV89, vdHvLM97, Rao95], involving a combination of temporal, modal or dynamic logics. Halpern et al. [HV89], for example, studies the complexity and axiomatisations of linear and branching-time temporal logics combined with multi-modal S5 (representing knowledge). The core of the KARO framework [vdHvLM97] centres around a combination of dynamic logic plus multi-modal S5/KD45. BDI logics [Rao95] use linear or branching-time temporal logics plus modal logics KD45 for belief and KD for desires and intentions.

In the Logic and Computation Group we are interested in proof methods for these complex combinations of logics. In particular we are extending the resolution methods developed for linear-time temporal logics [Fis91] to particular combinations of the above logics.

The resolution method centres round three main steps:-

- translation to normal form (essentially implications known as *rules* that hold at every moment in time);
- classical resolution between formulae that occur at the same moment in time; and
- resolution between sets of formula that make φ always true with constraints that ensure φ is false at some point in the future.

In [DFW98] this method was extended to linear-time temporal logic plus S5 by extending the normal form to allow rules with right hand sides that are disjunctions of literals or modal literals (eg. Kl and $\neg Kl$). As there is no interaction between the modal and temporal components temporal and modal reasoning can be done separately. Modal resolution rules, such as the following, that captures the T axiom, $Kl \Rightarrow l$, are added to the existing temporal resolution rules.

$$\begin{array}{rcl} \mathbf{true} & \Rightarrow & A \vee Kl \\ \mathbf{true} & \Rightarrow & B \vee \neg l \\ \hline \mathbf{true} & \Rightarrow & A \vee B \end{array}$$

The method requires at most a single nesting of modal operators within rules, due to the presence of the axioms 4 and 5. It has been extended to linear-time temporal logic plus KD45 (belief) in [DF99b]. Tableau methods for multi-agent versions of these logics have been given in [WDF98].

Current work is to extend the resolution for linear-time temporal logics with multi-modal versions of knowledge and belief. Further, many of the systems considered incorporate interaction between logics. For example Halpern et al. study the interaction

$$K \bigcirc \varphi \Rightarrow \bigcirc K \varphi$$

which can be added to the axioms of linear time temporal logic and S5 to give systems known as synchrony and perfect recall. Unfortunately such logics in general become much more complex than those with no interactions. For the single agent case without interactions the complexity of validity for knowledge and time without interactions is PSPACE whereas for synchrony and perfect recall it is double-exponential time [HV89]. Care must be taken when considering interactions as in many cases adding interactions makes the logic become undecidable. Even for logics with decidable interactions proposing sound resolution rules for such systems presents few problems [DF99a], proving completeness is more difficult. Finally we are interested in examining if the resolution method in [Fis91] can be extended to dynamic logics to deal directly with core parts of the KARO framework and using it to prove simple properties.

References

[DF99a] C. Dixon and M. Fisher. Clausal Resolution for Logics of Interacting Time and Knowledge. (In preparation), 1999.

[DF99b] C. Dixon and M. Fisher. Clausal Resolution for Temporal Logics of Belief. (Submitted), 1999.

[DFW98] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[Fis91] 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.

[HV89] J. Y. Halpern and M. Y. Vardi. The Complexity of Reasoning about Knowledge and Time. I Lower Bounds. *Journal of Computer and System Sciences*, 38:195–237, 1989.

[Rao95] A. S. Rao. Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics. In M. Wooldridge, K. Fischer, P. Gmytrasiewicz, N. R. Jennings, J. P. Müller, and M. Tambe, editors, *IJCAI-95 Workshop on Agent Theories, Architectures, and Languages*, pages 102–118, Montréal, Canada, August 1995.

[vdHvLM97] W. van der Hoek, B. van Linder, and J.J. Ch. Meyer. An integrated modal approach to rational agents. Technical Report UU-CS-1997-06, Department of Computer Science,Utrecht University, Centrumgebouw Noord,Padualaan 14, De Uithof,3584 CH Utrecht, 1997.

[WDF98] M. Wooldridge, C. Dixon, and M. Fisher. A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief. *Journal of Applied Non-Classical Logics*, 8(3):225–258, 1998.

[WJ95] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Composing Re-usable Synthesis Methods through Viewpoints

Jutta Eusterbrock
Department of Computing Science, University of Aberdeen
Aberdeen AB24 3UE, Scotland, UK
Email: jeusterb@csd.abdn.ac.uk

The focus of attention is on the reuse-oriented interplay between logic-based meta-level synthesis on the one hand and various multi-paradigm knowledge resources, ie. heterogeneous data, domain-specific knowledge and heuristic problem solving knowledge, located across distributed sources on the other. Many designers and users realise that information and knowledge in distributed software applications, databases and libraries as on the Internet provide valuable resources that can be reused for various constructive tasks like scheduling, component-based software and hardware design or planning. Recent demand from users unwilling to "surf the net" for long shed light on alternative solutions for retrieving and composing information items. The KRAFT project [Gra97] aims to develop a combination of database and artificial intelligence technology to allow users to find and exploit knowledge available on the Internet. Relieving the users from translating high-level constructive design specifications into intricate detailed queries and taking care of connection constraints requires the use of knowledge-based synthesis systems.

Methods for synthesis and associated systems should be like mathematical theorems in the sense that they state the assumptions which yield correctness, completeness and efficiency, and once methods have been deployed, they are *reusable* across various domains, can be seamlessly combined and never have to be recoded manually. Moreover, experimental studies have produced evidence that effective synthesis of solutions for practical problems requires the application of a broad range of detailed technics and knowledge – like efficient data encodings, comprehensive domain-specific and problem solving knowledge – to tackle exponential search complexities. The SEAMLESS multi-layer architecture [Eus96] is intended to provide guidelines for (1) the partitioning of synthesis fragments like types, synthesis knowledge, construction and learning methods into reusable units, and (2) the sound composition of units into effective domain-specific synthesis systems. Three layers are distinguished: domain knowledge, generic abstractions, distributed knowledge sources, each layer uniformly divided into the three levels, data types, declarative and executable specifications. Generic theories are sorted specifications. Sorts and predicates are intended to capture generic problem solving fragments; soundness axioms constrain their semantics. Generic tasks are specified through pre- and postconditions on sorted variables and additional constraints like efficiency. Generic methods are executable specifications (meta-interpreters) for the synthesis of solutions to instances of generic tasks, similar to proof plans [Bun88] and correct with respect to associated generic theories. They are defined in terms of "open" (undefined) function and predicate symbols, introduced by the associated generic theory. Open predicates are classified as either "hard" or "soft". Definitions for hard predicates have to be provided in order that a generic method terminates, definitions for soft predicates are optional and can speed up construction processes. *Logical viewpoints* (VPs) are independent intermediate distributable components which "lift" various kinds of object-level knowledge into a form suitable for constructive generic reasoning. That means, VPs implement generic theories and provide domain-specific computational definitions for the open symbols so that soundness axioms are preserved.

Hence, putting logic-based synthesis into practice involves providing construction knowledge and synthesis objects by precise logical specifications. What is most important to deal with intractable complexities is that domain-specific knowledge, such as solved cases, has to be operationalised. Finally, deductive results have to be communicated back in a user-friendly way, for example, by graphical support. Traditionally, synthesis systems are devised as stand-alone approaches. They rely on the user to pass information to the system as and when it needs to perform certain deductions, though this is a tedious and difficult task and guidelines are missing. Current Internet technologies make reuse oriented cooperative synthesis a viable option as they allow the loose coupling of distributed systems in such a way that they keep their autonomy, eg. based on a Blackboard architecture. Knowledge sharing is a central issue when building cooperative systems. User requirements and domain models tend to be hard to capture as what can be expressed by a logical formalism doesn't match the common representation means, eg. entity relationship diagrams or examples. There are several efforts on semantics-based information or knowledge sharing and inter-operable systems. Agent-based communication mandates a universal communication language in which inconsistencies and notational variations are being eliminated. Establishing communication between multi-paradigm components which range from conventional databases to meta-level synthesis systems and user-provided domain knowledge through knowledge sharing based on a syntactical variant of any plain formalism is complicated, if at all possible. Communication between a synthesis system and a knowledge resource needs some more subtle interaction rather than mere translation into a common format and polishing up notational variants. This talk will propose to adopt the viewpoint concept to make multi-paradigm resources accessible for meta-level synthesis based on formal methods. More precisely, invoking a viewpoint shall provide (1) physical access to distributed resources (2) procedures for computing values for sorted variables of the abstract type in terms of object-level expressions, eg. translation of program structure into graph-terms (3) an interface to object-level resources in terms of the abstract functions (4) domain-specific knowledge lifted into its meta-level representation, such that soundness axioms are preserved. Given such a viewpoint as dynamically added context, uniform meta-level representations of various kinds of domain-specific knowledge can be derived through high-level querying during run-time.

References

[Bun88] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. CADE'88*, pages 111–120. LNCS 310, Springer, 1988.

[Eus96] J. Eusterbrock. A multi-layer architecture for knowledge-based system synthesis. In *Proc. ISMIS'96*, pages 582–592. Springer Verlag, 1996.

[Gra97] P. Gray and et al. KRAFT: Knowledge fusion from distributed databases and knowledge bases. In R. Wagner, editor, *Proc. DEXA'97*, pages 682–691. IEEE Press, 1997.

Agents and Non-Classical Logics

Michael Fisher
Logic and Computation Group, Department of Computing and Mathematics
Manchester Metropolitan University, Manchester M1 5GD, U.K.
Email]: `M.Fisher@doc.mmu.ac.uk`
WWW [personal]: `www.doc.mmu.ac.uk/STAFF/M.Fisher`
WWW [group]: `www.doc.mmu.ac.uk/RESEARCH/LoCo`

Rather than expanding on one particular topic, in this abstract I will provide an outline of a range of activities concerning automated reasoning that I am currently involved in. Much of the work described below involves collaboration with other members of the Logic and Computation group at MMU.

Clausal Resolution for Temporal Logics

For several years, we have been developing clausal resolution methods for discrete linear temporal logics [Fis91, Dix96, Dix98, FDP98]. Our current work in this area concerns

- the refinement and efficient implementation of the basic proof methods [FD97, DF98], and
- extension of the approach, i.e. the use of a particular variety of normal form [Fis97a], to more expressive temporal logics such as temporal μ-calculus[1] and branching-time temporal logics [BDF99].

In addition, we have begun to investigate verification support for powerful first-order temporal logics via the use of proof planning techniques.[2]

Combinations of Temporal and Modal Logics

Combinations of temporal and modal logics are widely applicable within both computer science and artificial intelligence. In particular, temporal logics of knowledge and temporal logics of belief are commonly used in the specification and analysis of distributed, dynamic and multi-agent systems. As these are our main application areas (e.g. [FW97]), we are keen to develop appropriate proof methods for these logics[3]. Our starting point has been the extension of the above resolution method to these complex logics, and this has resulted in:

- resolution-based proof methods for temporal logics of knowledge [DFW98] and belief [DF99b];
- both resolution and tableaux-based methods for these logics dealing with certain interaction axioms [DF97, DF99a].

In addition, it has become clear that the normal form we have developed provides a general, and useful, mechanism for proof in combined logics [HFDS99].

[1] See `http://www.doc.mmu.ac.uk/RESEARCH/LoCo/mures.html`
[2] See `http://www.doc.mmu.ac.uk/RESEARCH/LoCo/mfotl.html`
[3] See `http://www.doc.mmu.ac.uk/RESEARCH/LoCo/pmtlkb.html`

Agent-Based Proof Methods

In addition to developing appropriate proof methods for formal methods in agent-based systems, we are also utilising agent-based systems in the *implementation* of mechanised reasoning. In particular, our recent work concerns

- an agent-based model of classical theorem-proving [Fis97b], together with its refinement and implementation [Kam98], and,
- an agent-based approach to proof planning [FI98].

This work is part of our longer term goal to develop dynamic societies of proof agents, and apply them to large and complex problems.

References

[BDF99] A. Bolotov, C. Dixon, and M. Fisher. Clausal Resolution for CTL*. (Submitted), 1999.

[DF97] C. Dixon and M. Fisher. Tableaux for Synchronous Systems of Knowledge and Time with Interactions. In *Proceedings of Scandinavian Conference on Artificial Intelligence (SCAI).* IOS Press, 1997.

[DF98] C. Dixon and M. Fisher. The Set of Support Strategy in Temporal Resolution. In *Proceedings of Fifth International Workshop on Temporal Representation and Reasoning.* IEEE Computer Society Press, 1998.

[DF99a] C. Dixon and M. Fisher. Clausal Resolution for Logics of Interacting Time and Knowledge. (In preparation), 1999.

[DF99b] C. Dixon and M. Fisher. Clausal Resolution for Temporal Logics of Belief. (Submitted), 1999.

[DFW98] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[Dix96] C. Dixon. Search Strategies for Resolution in Temporal Logics. In *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE).* LNCS, Springer-Verlag, 1996.

[Dix98] C. Dixon. Temporal Resolution using a Breadth-First Search Algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[FD97] M. Fisher and C. Dixon. Guiding Clausal Temporal Resolution. In *Proceedings of the Second International Conference on Temporal Logic (ICTL).* (To appear).

[FDP98] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. (Submitted), 1998.

[FI98] M. Fisher and A. Ireland. Towards a Science of Reasoning through Societies of Proof Planning Agents. In *CADE-15 Workshop on "Using AI Methods for Deduction"*, 1998.

[Fis91] M. Fisher. A Resolution Method for Temporal Logic. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, 1991. Morgan Kaufman.

[Fis97a] M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *Journal of Logic and Computation*, 7(4):429–456, August 1997.

[Fis97b] M. Fisher. An Open Approach to Concurrent Theorem-Proving. In J. Geller, H. Kitano, and C. Suttner, editors, *Parallel Processing for Artificial Intelligence*, volume 3. Elsevier/North-Holland, 1997.

[FW97] M. Fisher and M. Wooldridge. On the Formal Specification and Verification of Multi-Agent Systems. *International Journal of Cooperative Information Systems*, 6(1), January 1997.

[HFDS99] U. Hustadt, M. Fisher, C. Dixon and R. A. Schmidt. Normal Forms and Proof in Combined Modal and Temporal Logics. (In preparation), 1999.

[Kam98] H. Kampmann. Implementing Multi-Agent Theorem-Proving. MSc Thesis, Department of Computing and Mathematics, Manchester Metropolitan University, 1998.

Description Logics with n-ary Relations

Enrico Franconi and Ian Horrocks
Dept. of Computer Science, Univ. of Manchester
Manchester M13 9PL, UK
{franconi | horrocks}@cs.man.ac.uk
http : //www.cs.man.ac.uk/{∼franconi | ∼horrocks}

In this abstract we will briefly mention the research work which has been done at the University of Manchester on recent advances in Description Logics (DLs), namely the $\mathcal{DLR}$ generalisation of DLs towards n-ary relations. $\mathcal{DLR}$ was first introduced by [DL99] as a means for encoding expressive semantic data models for information systems such as extended Entity Relationship. It turns out that $\mathcal{DLR}$ elegantly generalises standard DLs, by allowing formulae referring to n-ary relations, not just to binary relations (i.e., roles). Still, the flavour of the obtained language is a modal logic, in the sense that formulae state properties holding in classes of worlds, determined by means of possibly complex paths constructed from n-ary relations. As a special case of the language, we again obtain a standard DL or a standard normal modal logic. We present here the first order fragment of the language introduced in [DL99]. We have chosen to limit the expressivity to first order since we are looking for a language implementable with the current technology. In particular, we refer to the current academic implementations of expressive DLs, namely the systems FaCT [Hor98] and iFaCT (FaCT extended with inverse relations). It has been recently demonstrated [HPS99] that the logic we are considering here allows for the implementation of sound and complete reasoning algorithms that behave quite well both in realistic applications and systematic tests.

The semantics of $\mathcal{DLR}$ rely on a model theoretic *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, consisting of a set $\Delta^{\mathcal{I}}$, called the *domain*, and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each relationship R of arity n a subset $R^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, such that the conditions in column 2 of Table 1 are satisfied (where A is an atomic concept, P is an atomic relationship, C and D are concepts, R and S are relationships, $\$i$ denotes the ith component of an n-ary relationship, and k is a non-negative integer).

It turns out that first order $\mathcal{DLR}$ theories can be encoded into an expressive DL theory, equipped with generalised axioms, inverse relationships, and graded modalities (i.e., qualified number restrictions) [DL99]. A $\mathcal{DLR}$ theory consists of a finite set of axioms of the form $C \sqsubseteq D$ and $R \sqsubseteq S$, where C and D are concepts, and R and S are relationships of equal arity. Such a theory can be encoded in an equisatisfiable DL theory using the encodings described in column 3 of Figure 1 (where C_R is a concept corresponding to the $\mathcal{DLR}$ relationship R, and U_i is a functional role representing the ith component of a $\mathcal{DLR}$ relationship). Note that $\mathcal{DLR}$ relations are reified as DL concepts, and the components of the relationship represented as functional roles. While the encoding obviously relies on the ability of the DL to reason with inverse roles, another less obvious consequence is that this DL does not have the finite model

$\mathcal{DLR}$ syntax	$\mathcal{DLR}$ semantics	DL encoding
$\top_{\mathcal{C}}$	$\top_{\mathcal{C}}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$\top_{\mathcal{C}} \quad (\top_{\mathcal{C}} \sqsubseteq \top)$
A	$A^{\mathcal{I}} \subseteq \top_{\mathcal{C}}^{\mathcal{I}}$	$A \quad (A \sqsubseteq \top_{\mathcal{C}})$
$\neg C$	$\top_{\mathcal{C}}^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\neg C$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$C \sqcap D$
$\exists[\$i]\mathsf{R}$	$\{d \in \top_{\mathcal{C}}^{\mathcal{I}} \mid \exists\langle d_1,\ldots,d_n\rangle \in \mathsf{R}.d_i = d\} \leqslant k\}$	$\exists U_i^{-}.C_R$
$\leqslant k[\$i]\mathsf{R}$	$\{d \in \top_{\mathcal{C}}^{\mathcal{I}} \mid \sharp\{\langle d_1,\ldots,d_n\rangle \in \mathsf{R} \mid d_i = d\} \leqslant k\}$	$\leqslant k U_i^{-}.C_R$
$\top_n$	$\top_n^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$	$\top_n \quad (\top_n \sqsubseteq \top)$
P	$\mathsf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$	$C_P \quad (C_P \sqsubseteq \top_n)$
$\neg\mathsf{R}$	$\top_n^{\mathcal{I}} \setminus \mathsf{R}^{\mathcal{I}}$	$\neg C_R$
$\mathsf{R} \sqcap \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \cap \mathsf{S}^{\mathcal{I}}$	$C_R \sqcap C_S$
$\$i_n : C$	$\{\langle d_1,\ldots,d_n\rangle \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$	$\top_n \sqcap \forall U_i.C$

Table 1: $\mathcal{DLR}$ concepts and relationships, their semantics and their DL encodings

property. New techniques have been studied which extend the algorithm of the FaCT system with blocking strategies that allow complete reasoning with such an expressive DL [HS99].

The $\mathcal{DLR}$ Description Logic was designed such that it is able to capture database schemata expressed in the most interesting Semantic Data Models and Object-Oriented Data Models [DL99, FS99]. For that purpose, the first order fragment suffices. The most common conceptual data model for database and information system design is the Entity-Relationship (ER) semantic data model. [DL99] refers to an extended version of the basic ER model, called EER, including taxonomic relationships, arbitrary boolean constructs, and entity definitions by means of both necessary and sufficient conditions and, more generally, by means of generalised axioms. It is shown how a schema expressed in an EER conceptual data model can be expressed in a $\mathcal{DLR}$ theory – whose models correspond with legal database states of the EER schemata – allowing for reasoning services such as satisfiability of a schema or the computation of a logically implied formula.

Various experiments have been done in the context of the European ESPRIT-4 Long Term Research project "Foundations of Data Warehouse Quality" (DWQ), using EER schemata as a source for testing the efficiency of reasoning with $\mathcal{DLR}$ theories in realistic domains. Previous approaches [CDR98] based on the translation of first order $\mathcal{DLR}$ theories into the basic $\mathcal{ALC}$ DL (in correspondence with multi-modal $\mathcal{K}$) introduce very large numbers of axioms, and have proved impractical. Using iFaCT and a slightly simplified version of the above encoding, promising initial results have been obtained with a $\mathcal{DLR}$ theory representing (fragments of) extended real world database schemata. Work is in progress to extend the algorithm implemented in iFaCT in order to allow a complete encoding of $\mathcal{DLR}$ [HST99]).

References

[CDR98] D. Calvanese, G. De Giacomo, and R. Rosati. A note on encoding inverse roles and functional restrictions in $\mathcal{ALC}$ knowledge bases. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 69–71. CEUR, May 1998.

[DL99] G. De Giacomo and M. Lenzerini. Description logic framework for information integration. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13. Morgan Kaufmann, 1999.

[FS99] Enrico Franconi and Ulrike Sattler. A data warehouse conceptual data model for multidimensional aggregation: a preliminary report. *AI*IA Italian Association for Artificial Intelligence journal*, 1999. To appear.

[Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 636–647. Morgan Kaufmann Publishers, San Francisco, CA, June 1998.

[HPS99] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 1999. To appear.

[HS99] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 1999. To appear.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. Submitted, 1999.

Cocktail:
A Tool for Deriving Correct Programs

Michael Franssen
Co-operation Centre Tilburg and Eindhoven Universities
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
E-mail mfranssen@usa.net

Introduction

At present, most tools for correct programming support functional or logical programming paradigms. However, in practice, most programs are written using procedural languages (including object-oriented languages) like Pascal, C/C++ or Java. Tools for procedural languages usually provide only syntactical support.

The essence of algorithms in procedural languages can be expressed in Dijkstra's guarded command language (GCL), for which there is a formal basis that allows semantical support. In fact, programs written in GCL can be *derived* from their specification, as is shown in [Dij76, Kal90]. So far, there are no tools that support this approach to programming.

The goal of Cocktail is to provide semantical support for deriving programs from their specifications. The system is intended for programmers in an educational setting. Cocktail is mainly practice-oriented to increase its usability for the intended users. Also, the system is set up to be extensible, flexible and easy to experiment with.

Cocktail is written entirely in Java and provides a modern graphical user interface. Proofs are graphically represented in Fitch-style natural deduction and can be manipulated by drag-and-drop operations.

Logical foundation

To ensure a firm logical foundation of the system, we use a typed lambda calculus for the logical parts of our system. Typed lambda calculi are suitable for interactive theorem proving, as is shown by systems like Coq, LEGO and Yarrow (see [Coq97]). Instead of using a fixed calculus, we use a logical framework of typed lambda calculi: the framework of Pure Type Systems (PTSs). This approach has several advantages. Firstly, it allows us to easily increase the expressive power of the logical foundation of the system at a later stage, without having to re-implement the entire logic. Secondly, we can use a type checking algorithm to verify the correctness of constructed proofs, thereby avoiding errors being introduced when the theorem prover gets large. Thirdly, we can communicate our proofs to other proof systems, since lambda terms are standardized proof objects.

On the other hand, typed lambda calculi are usually used for higher order logics, which are not (yet) needed for our system. During the derivation of a program, many first-order theorems have to be proved, most of which are relatively simple. Since for first order theorems good automation is possible, it is desirable to include an automated theorem prover (ATP) in the system. However, ATPs are usually based on other formalisms and use other techniques to prove theorems than interactive theorem provers.

Embedding First-Order Tableaux in a Pure Type System

In Cocktail, automated and interactive theorem proving are combined as follows: Firstly, an extension, introduced in [Laa97], to the standard PTSs is used to faithfully model first order logic. Secondly, a tableaux based theorem prover is implemented that directly uses the formulas from the extended PTS. Since we use a faithful model of first order logic, no problems will occur during the construction of the semantic tableau. Thirdly, the tableau constructed by the ATP is translated into a lambda-term of the underlying logic. A precise description of the translation algorithm and the resulting system can be found in [Fra97].

Linking Hoare Logic with a Pure Type System

In the Hoare logic or axiomatic semantics of the guarded command language supported by Cocktail, programs are annotated with proofs and specifications. The proofs in the annotation provide enough information to check a posteriori if a constructed program does indeed meet its specification. Just as a simple type checking algorithm can ensure that a constructed proof (lambda-term) is correct, a simple program checking algorithm ensures an annotated program is correct. In this way the soundness of Cocktail depends only on this checking algorithm, and not on the whole system, which may get quite large.

Result

Cocktail is an experimental system for deriving correct programs, aimed at students. The underlying logic can easily be extended and changed, since a logical framework is used rather than a fixed logic. Many of the simple theorems are proved automatically. The graphical user interface provides a clear view of the proofs and allows intuitive construction of proofs.

References

[Coq97] Coq. The Coq proof assistant. In *URL:* `http://pauillac.inria.fr/coq/`, 1997.

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming.* Prentice-Hall International, 1976.

[Fra97] Michael Franssen. λP–: A pure type system for first order logic with automated theorem proving. Computing Science Report 97–15, Eindhoven University of Technology, 1997.

[Kal90] Anne Kaldewaij. *Programming: the derivation of algorithms.* Prentice-Hall international series in Computer Science. Prentice Hall, 1990.

[Laa97] T. Laan. *The Evolution of Type Theory in Logic and Mathematics.* PhD thesis, Eindhoven University of Technology, 1997.

A Project to Build Background Knowledge into Refinement Operators for Inductive Logic Programming

Alan M. Frisch*
Intelligent Systems Group
Dept. of Computer Science, Univ. of York
York, YO10 5DD, United Kingdom

Many methods for machine learning can be seen as searching a space of hypotheses for a good one. In the framework of inductive logic programming (ILP) the search space is often taken be a set of clauses with either a specialisation or generalisation operator for moving between the clauses. Such an operator is called a *downward refinement operator* if it specialises clauses—that is, from a clause it moves only to entailed clauses. Similarly, an *upward refinement operator* generalises clauses; from a clause it moves only to clauses that entail it.

Since its inception, the field of ILP has been centrally concerned with the use of background knowledge in induction. Yet, surprisingly, no serious attempts have been made to account for background knowledge in refinement operators, even though such operators are one of the most important, prominent and widely-used devices in the field. This paper describes a project whose goals are to develop and study refinement operators that take account of background knowledge. As such refinement operators are relative to a theory—the background knowledge—we refer to them as *theory refinement* operators.

Since entailment between clauses is undecidable, refinement operators are often based on subsumption, a decidable approximation to entailment.[2] This project follows suit. Background knowledge is incorporated into subsumption simply by building it into instantiation. This is accomplished by using the results of Frisch and Page Jr. [FP95], who have formulated a very general definition of instantiation with built-in theories and have shown that it generalises other formulations—including the ones commonly used in sorted and equational unification.

This definition of subsumption with background knowledge provides the gold standard for theory refinement: two clauses should be connected by a sequence of applications of the theory refinement operator if and only if one of the clauses subsumes the other (relative to the background knowledge). One of the main problems being addressed by this project is the development of theory refinement operators that have this and other desirable properties. The elaboration of these operators depends crucially on the kind of theory that is to be

*Email: frisch@cs.york.ac.uk. Phone: +44 1904 432745. Fax: +44 1904 432767. World-Wide Web: http://www.cs.york.ac.uk/~frisch. Many of the papers cited here are available at this website.

[2]The texbook by Nienhuys-Cheng and Wolf [NCW97] provides an excellent exposition of subsumption-based refinement operators.

incorporated. This project intends to consider a variety of classes of theories, ranging from narrow classes to broad classes.

The idea of building a theory into a system through its mechanisms for instantiation has been well explored and exploited in automated deduction (by Frisch [Fri91] and many others). I conjecture that it is possible for ILP systems to also benefit greatly through the exploitation of instantiation with built-in theories. Frisch and Page Jr. [FP90, PF92, Pag93] have already explored this idea in the context of learning by finding common generalisations. This project is now exploring the idea in the context of refinement.

As a first step [Fri99] this project has developed a downward refinement operator for clauses into which is built a simple form of background knowledge—a sort theory, which encodes taxonomic knowledge. The clauses in the space of refinements are sorted in that each variable is associated with a sort and the variable ranges over only elements in that sort. The background knowledge that is built into refinement is a theory stating how sorts are related and what objects belong to what sorts. The background knowledge is incorporated into subsumption, which underlies the downward refinement operator, in one simple way: by restricting attention to those substitutions that are well-sorted in that the term substituted for a variable denotes an object that belongs to the sort associated with the variable. The development of a sorted downward refinement operator based on this new subsumption relation follows the same lines used by Nienhuys-Cheng and Wolf [NCW97] to develop an unsorted refinement operator based on ordinary subsumption.

Theory refinement operators promise certain advantages over ordinary refinement operators. In particular, the search spaces generated by theory refinement do a better job of ordering the clauses according to their generality in terms of covering (that is, entailing) training examples. Since the relationship between generality and order in the search space provides the basis for the powerful heuristics employed by many ILP systems, the use of theory refinement may improve the effectiveness of ILP systems. Developing an understanding of how, and to what extent, this can be exploited in solving real-world induction problems is a major goal of this project.

References

[FP90] Alan M. Frisch and C. David Page Jr. Generalization with taxonomic information. In *Proc. of the Eighth National Conf. on Artificial Intelligence*, pages 755–761, Boston, MA, July 1990.

[FP95] Alan M. Frisch and C. David Page Jr. Building theories into instantiation. In *Proc. of the Fourteenth Int. Joint Conf. on Artificial Intelligence*, pages 1210–1216, Montreal, Canada, 1995.

[Fri91] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[Fri99] Alan M. Frisch. Sorted downward refinement: Building background knowledge into a refinement operator for incuctive logic programming. Under review, February 1999.

[NCW97] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming.* Springer-Verlag, 1997.

[Pag93] C. David Page Jr. *Anti-Unification in Constraint Logics: Foundations and Applications to Learnability in First-Order Logic, to Speed-up Learning, and to Deduction.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1993.

[PF92] C. David Page Jr. and Alan M. Frisch. Generalization and learnability: A study of constrained atoms. In Stephen H. Muggleton, editor, *Inductive Logic Programming*, chapter 2, pages 29–61. Academic Press, London, 1992.

Extending PVS' Real Analysis Library with Transcendental Functions

Hanne Gottliebsen
University of St Andrews
School of Mathematical and Computational Sciences
Fife, KY16 9SS, Scotland
e-mail: `hago@dcs.st-and.ac.uk`

We extend the existing real analysis library of PVS, developed by Bruno Dutertre [Dut96], to include support for transcendental functions. In doing so we use John Harrison's development of real analysis [Har98] in HOL as a guide to which definitions and theorems to implement and prove.

Reals in PVS

PVS includes the real numbers as part of the provided system. Together with the definition is a large collection of theorems about linear expressions of real numbers, including the usual lemmas about commutativity, associativity and cancellation rules for equations and inequations.

PVS also has an exponentiation function, that is restricted to take non-negative integer powers of real numbers.

Dutertre's development of real analysis includes theorems about limits of sequences of reals, continuity and limits of functions, and about differentiation. In all of these areas Dutertre has used standard definitions.

The Extension

In order to develop the transcendental functions as limits of certain power series, we first develop a theory of partial sums, then consider sequences of these to determine convergence criteria for series. Particularly useful for this extension is the theory of convergence of sequences already available in Dutertre's library.

In general our definitions and lemmas are equivalent to Harrison's, but as we use Dutertre's library as a basis, and not Harrison's more general approach using convergence nets, our implementation is not quite as extensive. However, for the domain of transcendental functions, the two implementations do correspond closely.

In doing this development in PVS, having a list of theorems from the HOL library to prove is like having a map of a route from A to B with only the main towns indicated - one still has to work out the route between each town. Similarly here one has to work out the proofs, usually by using standard mathematics textbooks such as [TF88, Apo74]. As we shall see in the example we can not always use the standard proofs due to lack of some definitions (eg. of integration). In such cases one has to come up with alternative proofs, possibly by consulting the appropriate HOL proof script.

Example

To prove $|x| < 1 \Rightarrow x^n \rightarrow 0$ as $n \rightarrow \infty$, standard mathematical texts use $\ln(x)$ to do a direct proof, where $\ln(x)$ is defined by

$$\ln(x) = \int_1^x \frac{1}{y}\, dy$$

However as integration is not provided in PVS' analysis library, and this theorem is needed in the development of the series to define ln without integration, the standard proof is obviously not useful in this case. Instead Harrison proved the following: if a sequence s is unbounded, ie. $\forall y\, \exists N\, \forall n \geq N\ :\ s(n) > y$, then the sequence $1/s(n)$ converges to 0. This together with the fact that if $0 < |x| < 1$ then

$$\left|\frac{1}{x^n}\right| \geq 1 + n(\tfrac{1}{x} - 1)$$

gives a proof of the convergence of x^n.

Conclusion

The development of power series is almost complete, only lacking proof of term-by-term differentiation of power series. Thus the next step is to prove convergence of the series used to define the transcendental functions, leading on to theorems about calculations involving transcendental functions and a definition of π as the least positive x such that $\cos(x) = 0$. On completion we hope to make the development available as a PVS library.

References

[Apo74] Tom M. Apostol. *Mathematical Analysis.* Addison-Wesley, Reading, Massachusetts, 1974.

[Dut96] Bruno Dutertre. Elements of Mathematical Analysis in PVS. In J. Grundy J. von Wright and J. Harrison, editors, *Theorem Proving in Higher Order Logics: 9th International Conference.* Springer-Verlag, 1996.

[Har98] John Harrison. *Theorem Proving with the Real Numbers.* Springer-Verlag, 1998.

[TF88] George B. Thomas and Ross L. Finney. *Calculus and Analytic Geometry.* Addison-Wesley, 1988.

The Dynamic Creation of Induction Rules

Jeremy Gow, Alan Bundy and Ian Green,
Institute of Representation & Reasoning,
University of Edinburgh, 80 South Bridge,
Edinburgh EH1 1HN, Scotland
e-mail: {jeremygo,bundy,img}@dai.ed.ac.uk

A key problem in automating proof by explicit induction is the construction of an appropriate induction rule. The difficulty lies in the infinite number of possible induction rules, and the significant effect the choice can have on the base and step case proofs. It can make the difference between success and failure for an inductive theorem prover.

Traditional induction selection techniques rely on a predetermined set of induction rules. This set consists of the *dual inductions* of the terminating recursive functions[1] known to the system [Ste88], or sometimes induction rules supplied by the user, as in [KBB96]. Some techniques combine induction rules from this set, e.g. [Wal94], thus defining a 'space' of induction rules from which a choice can be made.

One weakness of these approaches is that given a particular space of induction rules, there will be theorems which the theorem prover can only prove, or can prove more easily, using an induction which is not in this space. To create such induction rules, [Pro94] dynamically constructed them during the base and step case proof search. Our aim is to design a more powerful and efficient proof strategy, based on an alternative approach to constructing the induction rule.

Our strategy constructs an appropriate step case by using second-order meta-variables to represent the unknown induction terms, and then instantiates them during the step case proof. Kraan used this technique to select appropriate constructor style inductions from a prestored set [KBB96]. We intend to dynamically construct an induction rule by constructing this step case, constructing the other base and step cases, and then proving the rule valid. These other cases will contain meta-variables which may be instantiated during their proofs. We intend to extend the λ-*Clam* proof planner [RSG98] using this strategy.

The initial schematic step case has a single inductive hypothesis and no non-inductive ones, i.e. for a conjecture $\psi(x)$, it is of the form

$$\psi(\mathcal{A}(x)) \vdash \psi(\mathcal{B}(x))$$

where $\mathcal{A}$ and $\mathcal{B}$ are second-order meta-variables. If a more complex step case is required, we allow the proof to fail and use a collection of proof critics[2] to suggest a suitable revision of the step case. Possible changes are the inclusion of another inductive hypothesis, or the transformation of a failed case split in a step case proof into several corresponding step cases.

[1] where the induction order is identical to the function's computation order.
[2] other uses of proof critics in *Clam* are described in [IB96].

The step case proofs are guided by rippling, which is potentially non-terminating in the presence of meta-variables. Termination can be ensured by limiting the number of *speculative ripples* [KBB96], i.e. where only wave-fronts speculated for this step are rippled. But in general an arbitrary number of speculative ripples may be required. Therefore we initially set a low limit, but provide a proof critic that can recognise when a failed proof might succeed given another speculative ripple. The critic can increase the limit accordingly.

Our approach has a number of advantages over previous work on dynamically constructing induction rules, based on the lazy generation of induction hypotheses [Pro94]. For instance, lazy generation employs certain steps which correspond to speculative rippling, and there is a fixed limit on the number of such steps for a particular conjecture. Hence this approach fails when all solutions require more than this limit. As outlined above, our approach allows this limit to be increased on demand from the critic, so such solutions may be found.

Another disadvantage of lazy generation is its restriction to *destructor style* inductions, where the induction conclusions are of the same form as the conjecture. Under this restriction there are examples which require additional lemmas involving additional functions. For example, there is an problem where our approach constructs the following suitable step case based on a given lemma

$$\psi(l) \vdash \psi(append(l, x :: nil))$$

But with lazy generation this must be expressed as

$$l \neq nil, \psi(chop(l)) \vdash \psi(l)$$

where *chop* removes the last element of a list. The proof now requires additional functions *chop* and *last*, and an additional lemma relating these to the example. These must be provided by the user, reducing the automation of lazy generation. Unlike lazy generation, our approach can deal with both destructor and non-destructor style induction rules.

References

[IB96] 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.

[KBB96] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for synthesis and induction. *Journal of Automated Reasoning*, 16(1–2):113–145, 1996. Also available from Edinburgh as DAI Research Paper 729.

[Pro94] M. Protzen. Lazy generation of induction hypothesis. In Alan Bundy, editor, *12th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, Vol. 814, pages 42–56, Nancy, France, 1994. Springer-Verlag.

[RSG98] J.D.C Richardson, A. Smaill, and I.M. Green. System description: proof planning in higher-order logic with lambdaclam. In C. Kirchner and H. Kirchner, editors, *Proceedings of CADE-15*, volume 1421 of *Lecture Notes in Computer Science*. Springer Verlag, 1998.

[Ste88] A. Stevens. A rational reconstruction of Boyer & Moore's technique for constructing induction formulas. In Y. Kodratoff, editor, *The Proceedings of ECAI-88*, pages 565–570. European Conference on Artificial Intelligence, 1988. Also available from Edinburgh as DAI Research Paper No. 360.

[Wal94] C. Walther. Mathematical induction. In C. J. Hogger D. M. Gabbay and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 12, pages 122–227. Oxford University Press, Oxford, 1994.

Maslov's Class K Revisited

Ullrich Hustadt
Renate A. Schmidt
Department of Computing, Manchester Metropolitan University,
Chester Street, Manchester M1 5GD, United Kingdom
Email: {U.Hustadt, R.A.Schmidt}@doc.mmu.ac.uk
URL: http://www.doc.mmu.ac.uk/STAFF/{U.Hustadt, R.Schmidt}

Abstract. We give a new treatment of Maslov's class K in the framework of resolution. More specifically, we show that K and the class DK consisting of disjunction of formulae in K can be decided by a resolution refinement based on liftable orderings. We also discuss relationships to other solvable and unsolvable classes.

Maslov's class K [Mas71] is one of the most important solvable fragments of first-order logic. It contains a variety of classical solvable fragments including the Monadic class, first-order logic with two variables (FO^2), the initially extended Skolem class and the Gödel class [FLTZ93]. It also encompasses a range of non-classical logics, like a number of extended modal logics, many description logics used in the field of knowledge representation [HS98, FLTZ93, chap. 7], and some reducts of representable relational algebras.

For this reason practical decision procedures for the class K are of general interest. According to Maslov [Mas71] the inverse method provides a means to decide the validity of disjunctions of formulae in the class K. Although Kuehner [Kue71] noted in 1971 that there is a one-to-one correspondence between derivations in the inverse method and resolution, only in 1993 a decision procedure for a subclass of the dual of K based on a refinement of resolution is described by Zamov [FLTZ93, chap. 6]. His techniques are based on non-liftable orderings which have limitations regarding the application of some standard simplification rules and the completeness proof relies on a π-ordering. Regarding completeness, it is not clear whether the problems with π-orderings as discussed in [Wei94] are avoided.

In this paper we are concerned with the problem of deciding satisfiability for the dual class of K, which we call $\overline{\mathrm{K}}$, as well as the class $\overline{\mathrm{DK}}$ consisting of conjunctions of formulae in $\overline{\mathrm{K}}$. In the context of resolution-based decision procedures this class is of particular interest, since even ordering refinements of resolution do not prevent the growth of term and variable height. Most of the results in the literature on decision procedures based on ordered resolution consider classes where (i) every non-ground term contains all the variables of a clause and either (ii) all literals share the same variables, or (iii) it is possible to identify a literal either by polarity or maximal height that contains all the variables of a clause. Even unrestricted resolution, and factoring, preserve properties (i) and (ii). Then an atom ordering can be utilised to ensure that for all clauses in a derivation the maximal height of variable occurrences does not increase and the height of literals is bounded [dN96, FLTZ93, chap. 4]. Consequently, any derivation terminates. In the case (ii) and (iii), a selection function and an atom ordering guarantees termination and, in particular, terms do not grow [GdN99]. A decision procedure which allows term height growth, but not variable height growth, by means of a selection function is described in [HS98]. $\overline{\mathrm{K}}$ is a class where the mentioned approaches will not prevent the growth of the variable height.

In this paper we describe a resolution decision procedure based on liftable orderings for the classes $\overline{\mathrm{K}}$ and $\overline{\mathrm{DK}}$, thereby extending the result of Zamov. Our proof is situated in the resolution framework of Bachmair and Ganzinger [BG94, BG97]. An additional renaming transformation of certain problematic clauses allows for the embedding of the classes under consideration into a class for which standard liftable orderings ensure closure under resolution and factoring as well as termination.

References

[BG94] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[BG97] L. Bachmair and H. Ganzinger. A theory of resolution. Research Report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1997. Preliminary version of a chapter in the *Handbook of Automated Reasoning*, edited by J. A. Robinson and A. Voronkov.

[dN96] H. de Nivelle. *Ordering Refinements of Resolution*. PhD thesis, Technische Universiteit Delft, The Netherlands, October 1996.

[FLTZ93] C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Method for the Decicion Problem*, volume 679 of *LNCS*. Springer, 1993.

[GdN99] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. To appear in Proc. LICS'99, 1999.

[HS98] U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caferra and G. Salzer, editors, *First-order Theorem Proving—FTP'98: Proceedings of 2nd International Workshop*, pages 152–161, 1998.

[Kue71] D. G. Kuehner. A note on the relation between resolution and Maslov's inverse method. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, chapter 5, pages 73–76. Edinburgh University Press, 1971.

[Mas71] S. Ju. Maslov. The inverse method for establishing deducibility for logical calculi. In V. P. Orevkov, editor, *The Calculi of Symbolic Logic I: Proceedings of the Steklov Institute of Mathematics edited by I.G. Petrovskiĭ and S. M. Nikol'skiĭ, number 98 (1968)*, pages 25–96. American Mathematical Society, 1971.

[Wei94] C. Weidenbach. Minimal resolution. Research Report MPI-I-94-227, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

On the Relation of Resolution and Tableaux Proof Systems for Description Logics

Ullrich Hustadt
Renate A. Schmidt
Department of Computing, Manchester Metropolitan University, Chester Street, Manchester M1 5GD, United Kingdom
Email: {U.Hustadt, R.A.Schmidt}@doc.mmu.ac.uk
URL: http://www.doc.mmu.ac.uk/STAFF/{U.Hustadt, R.Schmidt}

Abstract. We investigate the relationship between resolution and tableaux proof system for the satisfiability of general knowledge bases in the description logic $\mathcal{ALC}$. We show that resolution proof systems can polynomially simulate their tableaux counterpart. Our resolution proof system is based on a selection refinement and utilises standard redundancy elimination criteria to ensure termination.

Recently a number of results concerning resolution decision procedures for subclasses of first-order logic have been obtained. The considered subclasses are expressive enough to encompass a variety of non-classical logics, in particular, description logics and extended modal logics. De Nivelle [dN98] describes a resolution decision procedure for the guarded fragment using a non-liftable ordering refinement. The guarded fragment is a generalisation of the restricted quantifier fragment corresponding to basic modal logic and allows for the embedding of a variety of extended modal logics and description logics [Grä98]. Expressions and knowledge bases of the description logic $\mathcal{ALC}$ can also be embedded into Maslov's class K and its subclasses One-Free [FLTZ93] and the class of DL-clauses [HS98]. Again, ordering refinements of resolution provide decision procedures for these classes. A non-standard translation into the Bernays-Schönfinkel class combined with resolution and arbitrary refinements provide decision procedures for the satisfiability of $\mathcal{ALC}$ expressions [Sch98]. This approach was adopted in the experiments of [HS97, HSW98]. Experiments using the standard translation and a combination of a first-order theorem prover augmented with a finite-model finder are described in [PP98].

The problem of empirical investigations based on competitive testing is the difficulty in identifying the major factors having a positive or negative influence on the performance of a theorem prover. As long as the theorem provers which are being compared follow different proof strategies this difference is likely to have a dominating effect on the overall performance. This has two consequences. One, we can say little about the other factors influencing the performance, for example, fundamental differences between the underlying proof systems or sophisticated redundancy elimination techniques used by the theorem prover. Two, while it is easy to find benchmark problems illustrating the superiority of one theorem prover it is just as easy to find benchmark problems showing the opposite. Therefore, it is always advisable to complement empirical investigations with a theoretical analysis of the relative proof and search complexity of the underlying proof systems. In the first case, the task is to determine whether a proof system $\mathcal{A}$ is able to polynomially simulate a proof system $\mathcal{B}$. This is to say, for any given theorem ϕ there is a function g, computable in polynomial time, mapping proofs of ϕ in $\mathcal{B}$ to proofs of ϕ in $\mathcal{A}$. In the second case, the task is to determine the relative size of the search space, that is the potential number of inference steps performed until a proof is found [PZ97].

In this paper we focus on the aspect of relative proof complexity of tableaux proof systems and resolution proof systems for the description logic $\mathcal{ALC}$ with general terminological sentences and ABox elements. This logic is of particular interest, since all tableaux proof systems presented in the literature require some form of blocking or loop-checking to force termination [BDS93, DDGM96, Hor97]. We describe a resolution proof system based on a selection refinement of resolution, instead of an ordering refinement, which provides a new resolution decision procedure for this logic. We show that this proof system is able to polynomially simulate tableaux proof systems for this logic. The technique for simulating blocking described in this paper can also be applied for obtaining other simulation results, for example, analytic modal KE tableaux proof systems or sequent proof systems for modal logics.

References

[BDS93] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. Aritificial Intelligence Research*, 1:109–138, 1993.

[DDGM96] F. M. Donini, G. De Giacomo, and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. In L. Padgham, *et al.*, editor, *Proc. DL'96,* AAAI Technical Reports Series WS-96-05, pages 107–110, 1996.

[dN98] H. de Nivelle. A resolution decision procedure for the guarded fragment. In C. Kirchner and H. Kirchner, editors, *Proc. CADE-15*, volume 1421 of *LNAI*, pages 191–204. Springer, 1998.

[FLTZ93] C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Method for the Decicion Problem*, volume 679 of *LNCS*. Springer, 1993.

[Grä98] E. Grädel. Guarded fragments of first-order logic: A perspective for new description logics? In *Proc. DL'98*, volume 11 of *CEUR Electronic Workshop Proceedings*, 1998.

[Hor97] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[HS97] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In M. E. Pollack, editor, *Proc. IJCAI'97*, pages 202–207. Morgan Kaufmann, 1997.

[HS98] U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caferra and G. Salzer, editors, *First-order Theorem Proving—FTP'98: Proceedings of 2nd International Workshop*, pages 152–161, 1998.

[HSW98] U. Hustadt, R. A. Schmidt, and C. Weidenbach. Optimised functional translation and resolution. In H. de Swart, editor, *Proc. TABLEAUX'98*, volume 1397 of *LNAI*, pages 36–37. Springer, 1998.

[PP98] M. Paramasivam and D. A. Plaisted. Automated deduction techniques for classification in description logic systems. *J. Automated Reasoning*, 20(3):337–364, 1998.

[PZ97] D. A. Plaisted and Y. Zhu. *The Efficiency of Theorem Proving Strategies*. Vieweg, 1997.

[Sch98] R. A. Schmidt. Decidability by resolution for propositional modal logics. To appear in *Journal of Automated Reasoning*, 1998.

Automated Reasoning for Dependable Computing

Andrew Ireland and Jamie Stark and Andrew Cook
Dependable Systems Group
Department of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh, EH14 4AS
Email: a.ireland@hw.ac.uk jamie@cee.hw.ac.uk ceeajc@cee.hw.ac.uk
Tel: 0131 451 3409
Web: http://www.cee.hw.ac.uk/~dsg/

Introduction

Our work focuses upon the development of computer assisted theorem proving techniques which enhance our ability to understand and reason about the behaviour of computer systems. The need for such techniques is justified by the ever increasing demand for highly *dependable computer systems*, in particular within the avionics and semiconductor (System-On-Chip) industries. One of the longer term aims of our work is to develop techniques which are closely aligned to the needs and capabilities of system designers. Below we summarise our current research activities.

Proof Critics and Robust Reasoners

Success in the search for a mathematical proof typically involves the analysis of failed proof attempts. It is this analysis, coupled with a strong expectation as to how a proof should proceed, which often holds the key to success. A *proof plan*[1] [Bun88] makes such "expectations" explicit by providing a representation of the common structure which defines a family of proofs. This common structure can be used to automate the search for particular proofs. *Proof critics* [Ire92] are an extension of proof plans which automate proof patching through the analysis of failed proof attempts. The ability to overcome failure in this way leads to greater *robustness* in terms of the search for proofs and represents a distinguishing feature of theorem provers which are built upon the notion of proof plans. The proof critics mechanism has been applied very successfully within the domain of mathematical induction [IB96a, IB96b, IB97] where many eureka steps, *e.g.* missing lemmas and generalizations, have been discovered automatically. Mathematical induction is required when reasoning about recursion or iteration, so it is essential when reasoning about the correctness of computer systems. An interactive version of the proof critics mechanism has been implemented [Jac96, IJR98]. This work has

[1]The work described here is supported in part by EPSRC grant GR/M45030 in collaboration with Edinburgh Artificial Intelligence and Glasgow Caledonian.

made a significant contribution in terms of harnessing the potential of proof plans for supporting interactive proof development. Interactive proof critics are currently being investigated more extensively at Napier University. Recently we have started to explore the use of proof critics within the teaching of elementary logic to undergraduate students at Heriot-Watt.

Dependable functionality

Most of the work on proof plans has been concerned with reasoning about functional and logic programs. Recently we have been investigating[2] the use of proof plans for reasoning about the behaviour of imperative programs. In particular we have focused on the search control issues associated with the discovery and verification of loop invariants. In terms of formal reasoning, the discovery of a loop invariant is typically seen as a eureka step. The strong connection between iteration and recursion has enabled us to transfer proof plans and critics designed for reasoning about functional programs into this new domain giving some promising results [IS97, SI98]. We are currently investigating the use of proof plans in the automation of the synthesis and transformation of imperative programs from formal specifications. For instance, given the Floyd-Hoare style specification $\{x = \mathcal{X} \wedge y = \mathcal{Y}\}C\{r = exp(\mathcal{X}, \mathcal{Y})\}$ and a loop invariant of the form $r = exp(\mathcal{X}, i)$, then we are able to automatically synthesize an instantiation for C, *i.e.* a While-loop program, together with its proof of correctness:

$\{x = \mathcal{X} \wedge y = \mathcal{Y}\}$
$r := 1;\ i := 0;$
while not $(i = y)$ **do**
begin
  $r := r * x;$
  $i := i + 1$
end
$\{r = exp(\mathcal{X}, \mathcal{Y})\}$

Dependable performance

Improving the efficiency of programs through transformation is an important and active area of research. Within the Parallel SML compiler project[3] at Heriot-Watt, we are using transformation techniques to increase the opportunities for exploiting parallelism. The transformations we are interested in are those which introduce *higher-order functions* (HOFs), *i.e.* functions which take functions as arguments. HOFs are closely related to generic parallel constructs and consequently are amenable to parallelism. We are investigating the use of the proof planning framework to verify the correctness of the transformations used within the Parallel SML compiler. In addition, we aim to dynamically synthesize new transformations on demand, using proof plans to constrain the search. For example, given the conventional definition of $subset(X, Y)$, *i.e.* expressed in terms of the membership of the elements of X within Y, then we aim to automatically synthesize an equivalent function which is "rich" in

[2]This work is supported by EPSRC studentship 96307451 associated with the second author.
[3]EPSRC grant GR/L42889: Parallelising Compilation of Standard ML through Prototype Instrumentation and Transformation.

HOFs, *e.g.*

$$foldr(\lambda h.\lambda t.(foldr(\lambda a.\lambda b.((h = a)\ or\ b), false, X)\ and\ t), true, Y)$$

References

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[IB96a] A. Ireland and A. Bundy. Extensions to a Generalization Critic for Inductive Proof. In M. A. McRobbie and J. K. Slaney, editors, *13th Conference on Automated Deduction*, pages 47–61. Springer-Verlag, 1996. Springer Lecture Notes in Artificial Intelligence No. 1104. Also available from Edinburgh as DAI Research Paper 786.

[IB96b] 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.

[IB97] A. Ireland and A. Bundy. Automatic Verification of Functions with Accumulating Parameters. Research Memo RM/97/11, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 1997. Submitted to a special issue of the Journal of Functional Programming on Theorem Proving and Functional Programming. Also to appear as an Edinburgh DAI Research Paper.

[IJR98] A. Ireland, M. Jackson, and G. Reid. Interactive proof critics. 1998. Submitted to a Special Issue of the Journal of Formal Aspects of Computing on User Interfaces for Theorem Provers. Available from Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/98/15.

[Ire92] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 92, St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 178–189. Springer-Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[IS97] A. Ireland and J. Stark. On the Automatic Discovery of Loop Invariants. Research Memo RM/97/1, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 1997.

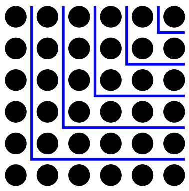
[Jac96] M. Jackson. HCI Techniques for Theorem Proving. Master's thesis, Department of Computing & Electrical Engineering, Edinburgh, 1996.

[SI98] J. Stark and A. Ireland. Invariant discovery via failed proof attempts. In *Proceedings of LOPSTR 98, University of Manchester Technical Report UMCS-98-6-1. Full version to be published LNCS Springer-Verlag.*, 1998. Also available from Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/98/2.

Automating Diagrammatic Reasoning*

Mateja Jamnik
The University of Birmingham, School of Computer Science
Birmingham, B15 2TT, UK
e-mail: `M.Jamnik@cs.bham.ac.uk`
WWW: `http://www.cs.bham.ac.uk/~mxj/`

Introduction



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

In my research I am looking into mathematical reasoning with diagrams. Human mathematicians often informally use diagrams when proving theorems. I investigate whether this kind of diagrammatic reasoning can be mechanised in a formal computer proof system. Diagrams have been used as an aid in mathematical reasoning as far back as the time of Euclid. They seem to convey information which is easily understood by humans. For example, it requires only basic secondary school knowledge of mathematics to realise that the diagram above is a proof of a theorem about the *sum of odd naturals*. I call such proofs diagrammatic proofs. My work is an investigation into formalising diagrammatic reasoning, a novel approach to mechanised mathematical reasoning. A concrete result of this investigation is a semi–automatic formal proof system, called Diamond, which allows a user to prove theorems of arithmetic using diagrams.

It is an interesting property of diagrams that allows us to "see" and understand so much just by looking at a simple diagram. Not only do we know what theorem the diagram represents, but we also understand the proof of the theorem represented by the diagram and believe it is correct.

Is it possible to simulate and formalise this sort of diagrammatic reasoning on machines? Or is it a kind of intuitive reasoning particular to humans that mere machines are incapable

*The research reported in this abstract was supported by Artificial Intelligence Dept. grant, University of Edinburgh, and the Slovenian Scientific Foundation supplementary grant. It has been carried out for my PhD research in the Department of Artificial Intelligence at the University of Edinburgh under the supervision of Prof. Alan Bundy and Dr. Ian Green. A good source of information about my work can be found in [JBG99]. For a more detailed account of my work, see [Jam99].

of? Roger Penrose claims that it is not possible to automate certain diagrammatic proofs.[2] I take his position as a challenge and try to capture the kind of diagrammatic reasoning that Penrose is talking about so that we will be able to emulate it on a computer. My motivation is not to discover diagrammatic proofs, but to study them in order to understand them better and be able to formalise them.

Diagrammatic Proofs and DIAMOND

Concrete rather than general diagrams are used to prove ground instances of a universally quantified theorem. The diagrammatic proof is constructed by applying geometric operations to the diagram. These operations are the inference steps of the proof. For example, the diagram at the beginning of this abstract, represents the instance $n = 6$. The operation which is applied to the diagram splits two adjacent sides from a square, call it lcut, and is applied six times, *i.e.*, recursively until the diagram is exhausted. A general schematic proof is extracted from a few ground instances of a proof (*e.g.*, for $n = 6$ and $n = 10$). It is represented as a recursive program, called proof, that consists of a general number of applications of geometric operations. When given a particular diagram, a schematic proof generates a proof for that diagram. In the example above, the schematic proof is:

$$\begin{aligned} \mathsf{proof}(n+1) &= [(\mathsf{lcut}, 1)],\ \mathsf{proof}(n) \\ \mathsf{proof}(0) &= [\,] \end{aligned}$$

To verify that the schematic proof produces a correct proof of the conjecture for each ground instance its correctness is checked in a theory of diagrams. The constructive ω-rule and schematic proofs are used to make a transition from concrete instances to a general argument about the diagrammatic proof.

The realisation of these ideas is a diagrammatic reasoning system DIAMOND. DIAMOND allows a user to interactively construct instances of a diagrammatic proof. It then automatically abstracts these into a general schematic proof and checks the correctness of this proof using an inductive theorem prover. Unlike other existing systems which use diagrams to construct essentially symbolic proofs, DIAMOND reasons with diagrams directly, so all the inference rules of a proof are diagrammatic.

Conclusion

Despite a popular view of logicians from the past century that diagrams cannot be used in formal proofs, I show the contrary. The general diagrammatic proof framework that I present in my work is a formalisation of diagrammatic reasoning. DIAMOND provides an environment in which *formal* diagrammatic proofs of mathematical theorems can be constructed.

References

[Jam99] M. Jamnik. *Automating Diagrammatic Proofs of Arithmetic Arguments.* PhD thesis, Division of Informatics, University of Edinburgh, 1999.

[JBG99] M. Jamnik, A. Bundy, and I. Green. On automating diagrammatic proofs of arithmetic arguments. *Journal of Logic, Language and Information*, Vol. 8, 1999. To appear.

[2] Roger Penrose presented his position in the lecture at International Centre for Mathematical Sciences in Edinburgh, in celebration of the 50th anniversary of UNESCO on 8 November, 1995.

On Truth, Knowledge, Strings, and Paradoxes

Manfred Kerber
The University of Birmingham, School of Computer Science
Birmingham, B15 2TT, England
e-mail: `M.Kerber@cs.bham.ac.uk`
WWW: `http://www.cs.bham.ac.uk/~mmk/`

One of the key issues in artificial intelligence is the representation and processing of knowledge. The selection of an appropriate formalism for doing so is a crucial step in building a usable system. In this selection process one tries to find an optimal balance between expressive power of the system and adequacy of the representation process on the one hand and efficiency and simplicity of the reasoning process on the other hand.

Many knowledge representation formalisms are built on top of some logical system. A powerful and well-understood system is classical first-order logic. Since it has some deficiencies, extensions and alternatives have been studied. In particular modal logics have been studied in great detail in recent years. In recent work I advocate a system that is more powerful but simpler than modal logic. It is a simple extension of classical first-order logic, namely first-order logic plus strings. Although known for quite a while and in spite of its positive features, it has not attracted much attention since it is easy to formulate self-referential paradoxes within this system. I have studied the reason for self-referential paradoxes and came up with a semantical condition on the connectives such that these paradoxes can be excluded. This leads to a powerful three-valued system that makes it possible to use reasoning techniques for classical first-order logic.

First-order logic with strings is a conservative extension of first-order logic in which strings as in $short(\text{“}cat\text{”})$ stands for $short(pair(:c, pair(:a, :t)))$. With such a construction it is possible to define syntactic and semantical predicates about the language itself. In particular it is possible to define the syntax of the language within the language itself. Details of such a construction can be found in [GN87, Chapter 10]. Although it is intended that strings and the objects they stand for are closely related, we must carefully distinguish between them. There is an intuitive difference between $\mathsf{long}(\mathsf{John})$ and $\mathsf{long}(\text{“}\mathsf{John}\text{”})$ (where long is assumed to be a polymorphic predicate symbol that is true for persons over 195cm and strings consisting of at least eight characters.)

In such a setting it is easy to express a problematic concept like the liar sentence:

- This sentence is false.
  $\mathbf{L} :\equiv \neg\mathsf{True}(\text{“}\mathbf{L}\text{”})$

If we assume a standard first-order semantics there is no problem with the sentence above, since strings like “**L**” are ordinary ground terms in the language and have nothing to do with their counterparts like **L**. However, as such they are not particularly useful and do not capture the intended meaning. Of course we would like the predicate True to mean truth. Tarski [Tar36] gave a famous definition of truth:

$$\mathsf{True}(\text{``}\mathbf{A}\text{''}) \equiv \mathbf{A}$$

as axiom schema for arbitrary formulae $\mathbf{A}$. If we add this schema the liar sentence is turned into a paradox, since we get: $\mathbf{L} \equiv \neg\mathsf{True}(\text{``}\mathbf{L}\text{''}) \equiv \neg\mathbf{L}$, hence $\mathbf{L} \equiv \neg\mathbf{L}$.

The problem occurs since by a definition we introduce a contradiction, (it is not possible to assign any truth value to $\mathbf{L}$). Naive attempts to solve this problem by adding a third truth value fail since it is possible to express sentences like "This sentence is paradoxical or false", which gives rise to the same problem on the next higher level. None of the then three truth values can be consistently assigned to the sentence. This is due to the fact that the genuine reason for the occurrence of the paradoxes has not been studied and the idea of adding just another truth value can at best be an attempt to "bar monsters" as Lakatos put it [Lak76].

The reason for the occurrence of paradoxes is the possibility to define new predicates which can't have a truth value. If we assume that any function on the truth values is monotonous and fixpoint-free, we can convince ourselves that these problems cannot occur. Since $\neg$ and $\vee$ are of this kind in three-valued Kleene logic, it is possible to define a system which makes use of these connectives only (and $\wedge$, $\rightarrow$, and $\leftrightarrow$ which can be defined in $\neg$ and $\vee$) in which no "higher type" paradoxes are expressible and simple paradoxes like that of the liar sentence can be covered. I have recently developed a corresponding three-valued system, which allows to use a tableau calculus and more or less standard first-order theorem proving techniques. It includes a definition principle for predicate symbols. On the one hand it is very expressive since it allow for self-referential assertions (which are difficult to avoid in multi-agent contexts) and restricts the expressive power only insofar as it is not possible to speak about paradoxicality. On the other hand it is much simpler than the systems developed in [Kri75] and [BE87] and allows for an efficient mechanisation. For details see [Ker98, Ker99].

References

[BE87] Jon Barwise and John Etchemendy. *The Liar – An Essay on Truth and Circularity.* Oxford University Press, New York, USA, 1987.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, San Mateo, California, USA, 1987.

[Ker98] Manfred Kerber. On knowledge, strings, and paradoxes. In Jürgen Dix, Luís Fariñas del Cerro, and Ulrich Furbach, editors, *Proceedings of JELIA'98, Logics in Artificial Intelligence*, pages 342–354. Springer Verlag, Berlin, Germany, 1998. LNAI 1489.

[Ker99] Manfred Kerber. On truth, knowledge, strings, and paradoxes. submitted, 1999.

[Kri75] Saul Kripke. Outline of a theory of truth. *The Journal of Philosophy*, LXXII:690–716, 1975.

[Lak76] Imre Lakatos. *Proofs and Refutations.* Cambridge University Press, 1976.

[Tar36] Alfred Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia philosophia*, **1**:261–405, 1936.

Automated Reasoning and Practical Formal Methods

Helen Lowe
Dept of Computer Studies
Glasgow Caledonian University
e-mail: `H.Lowe@gcal.ac.uk`
WWW: `http://www.callnetuk.com/home/helenlowe/`

Gordon Reid
Division of Informatics
University of Edinburgh
e-mail: `Gordon.Reid@ed.ac.uk`
WWW: `http://www.dai.ed.ac.uk/~gordon/`

Within the Informatics community there is a growing recognition that, as software becomes more pervasive in our society, its reliability needs to be guaranteed more stringently than has previously been the case. This is most obvious in the area of safety-critical systems, if less so in mission and business critical fields.

In many countries legislators, recognising this need, are enacting guidelines and laws which are beginning to make adherence to strict methods of software development mandatory. It is not yet the case that the most formal of methods, involving mathematical correctness, are what is being called for but we are at least seeing some steps in the direction that proving system designs correct may eventually be required.

The view that proof is not only desirable but may soon be necessary is driving a renewed interest in theorem proving.

Validation and verification are time-consuming activities which existing theorem proving technologies have not easily been able to accomplish within the time and budget constraints of typical software projects. There is a gap between the needs of practical formal methods practitioners and the theorem proving tools available to fulfill these needs. If the typical software engineer is to be required to demonstrate the reliability of his or her work in a formal framework then this is a challenge to the automated reasoning community to provide the practical basis for the necessary technology.

There are, however certain criteria these tools must adhere to to be generally accepted by formal methods practitioners.

These are

- a high level of automation
- where automation is not possible interaction and explanation at a high level of abstraction is necessary

- usability by non-experts in theorem proving

The Mathematical Reasoning Group in the Division of Informatics, University of Edinburgh has, over the years, developed a number of "technologies" which may form the basis for such tools for practical formal methods.

Proof planning is a well-established technique for guiding the search for a proof and in particular inductive proof of the kind that is often required in formal methods. A proof plan is a high level description of a proof which acts as the specification of a tactic for an object-level theorem prover. This is similar to the way an abstract specification is a high level description of the design of a software system. This analogy may be a useful bridge in "selling" theorem proving to software engineers.

Within the proof planning paradigm several techniques aid the search for and discovery of a proof.

- Rippling
- Middle Out Reasoning
- Proof critics
- Interactive Proof critics
- Proof presentation and user-interface design

It is our intention to exercise our proof planning systems on examples from the practical formal methods literature in an effort to ascertain their strengths and weaknesses and to pinpoint areas for further work arising directly from the requirements of the formal methods community. We have chosen first of all to examine the Vienna Development Method as the recent acceptance of an ISO standard for VDM highlights its position as a widely accepted method. We have also formed a link with a major collaborative project on mechanised formal specification and verification, Prosper {`http://www.dcs.gla.ac.uk/prosper/`}.

Previous work with proof in VDM has lead to the development of several prototype theorem proving tools. These have been based on generic interactive theorem provers such as Isabelle, HOL, PVS and mural. It is obvious however that as yet none of these systems is of practical use to the ordinary software engineer using VDM. All require an intimate knowledge of theorem proving and the specific theorem prover and they are time-consuming and difficult to use.

Proof in VDM, if it is done at all, is still left to specialists who often end up doing it by hand. This however can still be beneficial as simply attempting a proof often highlights errors in the software specification. An automated proof plan, of the kind that our systems can produce, displays a level of rigour that may be suitable for showing the correctness or otherwise of a specification without the time-consuming effort of a fully formal proof. Once a plan is complete, with amendments to the specification if necessary, then a fully formal proof can be generated automatically.

Formal methods provides an avenue of research into the practical application of automated reasoning which may be beneficial to both areas.

Set Description Languages and Reasoning about Numerical Features of Sets

Hans Jürgen Ohlbach
Dept. of Computer Science, Kings College
Strand, London WC2R 2LS
email: ohlbach@dcs.kcl.ac.uk

Abstract Set description languages (SDL), for example description logics, can be used to specify sets and set-theoretic relationships between them. Mathematical programming, on the other hand, can be used to find optimal solutions for arithmetical equation and in-equation systems. In this paper a combination methodology is presented, which allows one to use numerical algorithms for reasoning about numerical features of sets specified with the description logic $\mathcal{ALC}$ and a few of its variants.

It is well known that formulae of propositional logic (PL) can be interpreted set-theoretically. Therefore a set of propositional axioms can be used to state set theoretic relationships between different sets. Numerical features of sets, in particular cardinality information, cannot be expressed in PL. In [OK98], Ohlbach and Koehler have presented *atomic decomposition* as a general technique for combining formal systems with a Boolean algebra component, i.e. propositional logic, as a set description language. This system is in particular suitable to extend mathematical programming methods with a propositional component in such a way that the mathematical programming system can be used to solve constraint systems about all kinds of numerical features of sets. The language used to express numerical features of sets uses propositional formulae embedded in *bridging functions* which map the sets to numbers. The cardinality function is one particular bridging function, but there may be others. The only requirements is that the numerical value $f(x \cup y)$ for two disjoint sets x and y can be computed by computing $f(x)$ and $f(y)$ first and then combining the results with some arithmetical function. The technique presented in [OK98] uses *atomic decomposition*, which is well known from Boolean algebra theory, to eliminate the set part from a combined axiomatization, and to turn the reasoning problem into a pure arithmetical problem.

Propositional Logic is not a very expressive set description logic. More expressive logics for reasoning about sets are Description Logics, such as $\mathcal{ALC}$ [SSS91, BBH96]. Unfortunately, the algebraic models behind $\mathcal{ALC}$ are no longer Boolean algebras, therefore the atomic decomposition technique can not be applied straightforwardly. However, I can show that with one extra *interaction axiom* $|\varphi| = 0 \Rightarrow |\exists\, r.\varphi| = 0$. (If the set φ is empty then the set of objects having some r-successors in φ must be empty too.) one can apply the atomic decomposition technique again. Due to lack of space, I cannot present the details here, but I want to illustrate the technique with an example.

Example The informal specification of the problem is: *for political polls concerning immigration laws, a number of at least 2300 U.S. people have to be surveyed, at least 1000 of* *them have to be 30 years or younger and 600 between 31 and 50. At least 15% of the surveyed people have to live in a state bordering Mexico and not more than 20% of those who are 51 or more must live in a state bordering Mexico. The prices of interviewing people in each age and region category are $7.50 for those younger than 30 and living in a state bordering Mexico and $6.90 for those who do not live in a state bordering Mexico, and so on.*

The sets of people to be interviewed are axiomatized in $\mathcal{ALC}$ with number-valued functions (*age* in this case). One possibility is:'

$$\begin{array}{lclllcl} 30^- & = & int \sqcap age \leq 30 & & bs & = & int \sqcap (\exists\, resident\text{-}in.bordering\text{-}state) \\ 30^+ & = & int \sqcap \neg(age \leq 30) \wedge age \leq 50 & & nbs & = & int \sqcap (\neg\exists\, resident\text{-}in.bordering\text{-}state) \\ 50^+ & = & int \sqcap \neg(age \leq 50) & & int & \neq & bordering\text{-}state \end{array}$$

where *int* denotes the set of people to be interviewed, etc. The arithmetic part of the specification can now be formulated using the notions introduced above:

$$|int| \geq 2300,\quad |30^-| \geq 1000,\quad |30^+| \geq 600,\quad |bs| \geq 0.15 * |int|,\quad |50^+ \sqcap bs| \leq 0.20 * |50^+|,$$
$$costs(bs \sqcap 30^-) = 7.50 * |bs \sqcap 30^-|,\quad costs(bs \sqcap 30^+) = 6.80 * |bs \sqcap 30^+|,\quad \ldots$$

The problem is to find numbers for the cardinalities of the sets such that $costs(int)$ is minimal. In the fist step one has to replace the defined concepts by their definitions, which is straightforward in this case. The set formulae to be decomposed are int, $age \leq 30$, $age \leq 50$ and $bs \stackrel{\text{def}}{=} \exists\, resident\text{-}in.bordering\text{-}state$ There are 6 atoms which are consistent with the above definitions of these notions:

$$\begin{array}{lclllcl} b30^- & = & int \sqcap bs \sqcap age \leq 30 & & b30^+ & = & int \sqcap bs \sqcap \neg(age \leq 30) \sqcap (age \leq 50) \\ b50^+ & = & int \sqcap bs \sqcap \neg(age \leq 50) & & & & \end{array}$$

together with $nb30^-$, $nb30^+$ and $nb50^+$, which are defined in the same way as above. Using these six atoms one can decompose the set terms into sets of atoms, and the functions operating on Boolean terms – in this example the cardinality function and the *costs* function – into new number valued variables. The result of all these transformations is

$$n_{b30^-} + n_{nb30^-} + n_{b30^+} + n_{nb30^+} + n_{b50^+} + n_{nb50^+} \geq 2300$$
$$n_{b30^-} + n_{nb30^-} \geq 1000, \qquad n_{b30^+} + n_{nb30^+} \geq 600$$
$$n_{b30^-} + n_{b30^+} + n_{b50^+} = 0.15 * (n_{b30^-} + n_{nb30^-} + n_{b30^+} + n_{nb30^+} + n_{b50^+} + n_{nb50^+})$$
$$n_{b50^+} \leq 0.2 * (n_{b50^+} + n_{nb50^+}),\quad c_{b30^-} = 7.50 * n_{b30^-} \qquad \text{etc.}$$

This is now a standard representation of an optimization problem with objective function $c_{b30^-} + c_{nb30^-} + c_{b30^+} + c_{nb30^+} + c_{b50^+} + c_{nb50^+}$ to be minimized. The solution is $c_{nb30^-} = 1000$, $c_{b30^+} = 600$, $c_{b50^+} = 140$, $c_{nb50^+} = 560$.

References

[OK98] H. J. Ohlbach and J. Koehler. How to extend a formal system with a Boolean Algebra component. In P.H. Schmidt and W. Bibel, eds, *Automated Deduction. A Basis for Applications*, Volume III, pp 57–75. Kluwer Academic Publishers, 1998.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[BBH96] F. Baader, M. Buchheit, B. Hollunder, Cardinality restricitions on concepts, Artificial Intelligence, Vol 88, pp. 195-213, 1996.

The Application of Model-Checking to Verification Problems in Telephone Switching Systems

Stephan Reiff and Muffy Calder
University of Glasgow, Department of Computer Science
Glasgow, G12 8QQ, Scotland
e-mail: `{sreiff,muffy}@dcs.gla.ac.uk`
WWW: `http://www.dcs.gla.ac.uk/{~sreiff/,~muffy/}`

Abstract

Model-checking is becoming very popular for automated reasoning about hardware and software. The input can be in any language provided there is a translation to transition systems or Kripke structures; properties are then expressed in a temporal logic.

Telephone switching systems are concurrent systems which are naturally described by a composition of interacting finite automata. We describe a model of one switching system in IO-Automata, express it in PROMELA (process meta language) and then formulate certain properties in LTL (linear temporal logic). Using the model-checker SPIN, we have attempted to verify the model with respect to some properties. A particular challenge is that even a simple switching system leads to state space explosion.

1 Introduction

Telephone switching systems have developed from manual switchboards to automated systems. The latter require control software, which is usually large, unstructured and not well documented (legacy software). On the other hand, increasing user demands are forcing the provision of extensions, usually in the form of features. Implementing features is not straight forward, as in addition to the desired interactions with the legacy system and other features, undesirable ones also occur. This is commonly known as the *feature interaction problem*. A typical example for an undesirable interaction is an inconsistency between *call waiting* and *call forwarding on busy*. The former requires to provide a call waiting tone, the latter tries to forward the call – both are triggered by an incoming call on a busy line.

To improve the provision of features in telephone switching systems, we are investigating methods to detect and resolve feature interactions. We are convinced that model-checking will provide results that can be used to improve operational and legacy systems as well as their development process. But to achieve this, the model needs to be an appropriate abstraction of the operational system, i.e. realistic enough to represent the system while being simple enough to be formally verified.

We propose such a model in Section 2 and give details of one component in the subsequent section. In Section 4 we comment on the implementation in PROMELA (PROcess MEta LAnguage) – the input language for the model-checker SPIN([Hol91]) and in section 5 show examples of properties that we wish to verify. In the conclusion we comment on the achievements so far, and discuss further directions.

2 A Telephone Switching System

In this section we give a high-level operational overview of a telephone switching system. As can be seen from figure (2) the system consists of three principal groups: the network, users and the operator. The operator interactions are mainly database enquiries and manipulations and do not concern us here.

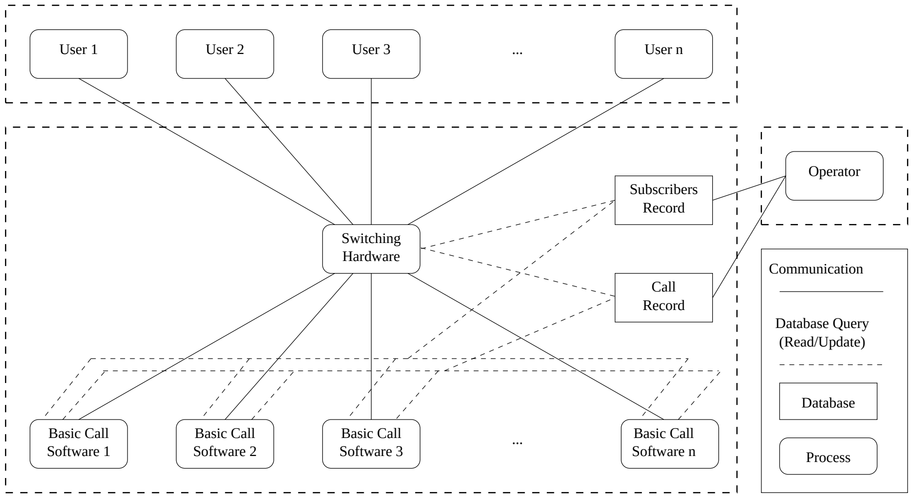


Figure 2: Overview of the System

The term 'user' will be used to describe points at which the network interacts with the world, such as terminal devices, trunks (connections to other networks) or any other communication equipment. For simplification, we only consider telephones as terminal devices and assume that there is only one switch.

The network consists of two database components (the call record and the subscribers record), which are used for control and billing purposes. Furthermore, we have a switching hardware process which passes messages between users and basic call software processes (and combinations thereof). Each user is assigned a basic call software process; they are only instantiated on demand and provide POTS (Plain Old Telephone System) call behaviour.

This model might seem rather more simplistic than a PSTN (Public Switched Telephone Network) but it does, for example, describe a PBX (Private Branch Exchange).

3 A Basic Call Software Process

We use IO-Automata to model the processes. An IO-Automata ensures that an input message is never blocked but allows us to place restrictions on when an output message is sent ([LT88]). Furthermore, we can compose such automata, allowing communication via their respective input and output messages.

Figure (3) shows the automaton for a basic call software process, where input messages are in italics, and all other messages are output messages. A message consists of two parts, an event and a numeric argument. The argument is usually an user identifier, or 0 when no argument is required. Note that once this automaton is placed in an environment there will be no non-deterministic choices, as choices are determined either by the received input actions or data from the databases. Nevertheless, any path through the automaton is possible. In the following example we describe a successful call to user 4 from another user (user 3 say).

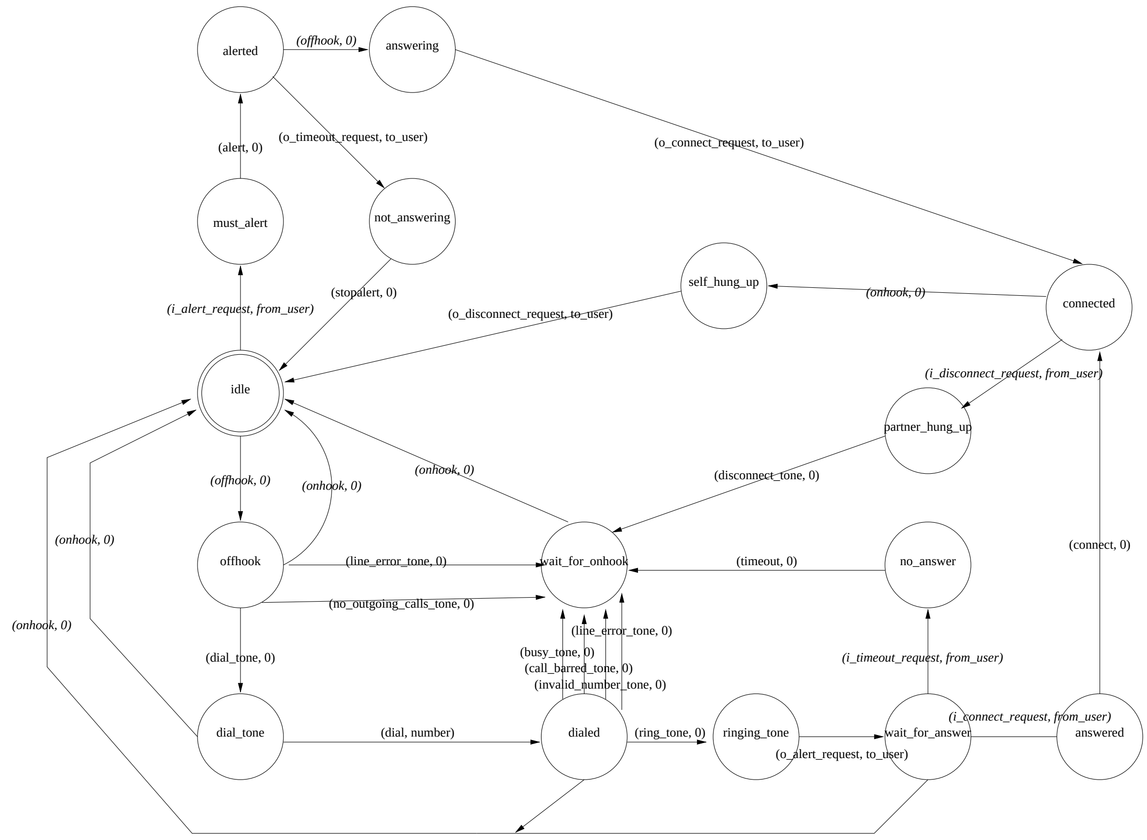


Figure 3: The Basic Call Software Process

We only consider the basic call software relating to the originating behaviour, ignoring any terminating behaviour.

Example 1 *The basic call software receives an $(offhook, 0)$ message to which it responds with $(dial\_tone, 0)$. Next a $(dial, 4)$ messages triggers the sending of $(ring\_tone, 0)$ and $(o\_alert\_request, 4)$. User 4 goes offhook which leads to an $(i\_connect\_request, 4)$ message being received, triggering $(connect, 0)$ being sent. Now the two users are connected. An $(onhook, 0)$ indicates that the connection is to be cleared down, triggering an $(o\_disconnect\_request, 4)$ message being sent. At this point the basic call software is in its originating state and a new call can be made.*

We note that this automaton does not describe the real behaviour in a PSTN. For example in the current BT network only the originator of a call can clear down the call (with exceptions, like 999). However, many PBXs already show symmetric behaviour.

4 The Promela Implementation

Implementing the above system in PROMELA is largely a matter of mastering the syntax correctly. Promela uses channels for communication. Each automaton is represented by a process, the number of arguments passed to each process is determined by the number of communication channels it needs (e.g. a basic call software process needs two uni-directional channels).

5 Verification of Some Properties

Properties of the system are expressed in LTL (Linear Temporal Logic) for automatic proofs in Spin. Here will give them only informally, as the LTL formulation can only be understood with respect to details of the model implementation.

Examples of properties we wish to verify are:

- If user A calls him/herself he/she should perceive a busytone.
- If there are no faults with the lines, user A is allowed to originate calls and user B is not busy, then eventually they can be connected.

These property might sound not very interesting, but we note that once a feature such as *call forwarding* is added to the system, they might become invalid.

6 Conclusions

Our long term goal is to reason about feature interactions in legacy systems with added features. As a first step we have constructed a model of an existing telephony system (possibly legacy) and investigated the use of model-checking techniques for analysis.

Next steps include adding features and a feature manager ([MM98]) to the model, derive new properties and prove those. These properties rank from if no feature is activated, then the system behaves exactly like POTS, to totally new behaviour that is expected of activated features. We want to detect unwanted interactions as well and resolve those (hence the feature manager). Generic properties (e.g. livelock or deadlock) are of no less interest, because they are often consequences of feature interactions. In fact they may be more useful, when dealing with systems where we do not have access to feature specifications, because they are but are independent of a particular features's expected behaviour.

The work reported here provides a basis for a prototype implementation, future feature development processes, and, hopefully, decision making at runtime when an interaction is detected.

Related work on exploring user views of telephone switching systems, as in [CM98] and general work on feature interaction shows the importance of finding solutions to the problem ([KB98], [DBL97] and others).

References

[CM98] M. Calder and A. Miller. Analysing a basic call protocol using Promela/XSpin. In *Proceedings* Spin*'98: Workshop on automata theoretic verification with the* Spin *model checker*, pages 169–181, Paris, France, November 1998.

[DBL97] P. Dini, R. Boutaba, and L. Logrippo, editors. *Proceedings of the* 4^{th} *International Workshop on Feature Interactions in Telecommunications Systems.* IOS Press (Amsterdam), June 1997.

[Hol91] G. J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall (New York), 1991.

[KB98] K. Kimbler and L. G. Bouma, editors. *Proceedings of the* 5^{th} *International Workshop on Feature Interactions in Telecommunications and Software Systems.* IOS Press (Amsterdam), September 1998.

[LT88] N. A. Lynch and M. R. Tuttle. An introduction to Input/Output Automata. November 1988.

[MM98] D. Marples and E. H. Magill. The use of rollback to prevent incorrect operation of features in intelligent network based systems. In K. Kimbler and L. G. Bouma, editors, *Proceedings of the* 5^{th} *International Workshop on Feature Interactions in Telecommunications and Software Systems*, pages 115–134, Lund, Sweden, September 1998.

Proof Planning in Higher-Order Logic with λ*Clam*

Julian Richardson

Proof planning is an approach to theorem proving which encodes heuristics for constructing mathematical proofs in a meta-theory of methods. The *Clam* system, developed at Edinburgh, has been used for several years to develop proof planning, in particular proof plans for induction.

It has become clear that many of the theorem-proving tasks that we would like to perform are naturally higher-order. For example, an important technique called middle-out reasoning uses meta-variables to stand for some unknown objects in a proof, to be instantiated as the proof proceeds. Domains such as the synthesis and verification of software and hardware systems, and techniques such as proof critics, benefit greatly from such middle-out reasoning. Since in these domains the meta-variables often become instantiated with terms of function type, reasoning with them is naturally higher-order, and higher-order unification is a vital tool.

One of our motivations for studying induction is that it is an important technique for reasoning about programs. Reasoning about programs (functions, predicates) is naturally higher-order, and although it is possible to reason about such objects in a first-order meta-theory, in order to do so we would have to modify our existing *Clam* methods to deal with proper treatment of lambda bindings, beta reduction etc. The reasoning is much clearer if we use higher-order logic as the meta-theory.

λ*Clam* is a higher-order prototype of *Clam*, written in λProlog, a higher-order version of Prolog. The meta-theory is based on higher-order logic, which simplifies reasoning about higher-order objects and allows many of the proof plans developed in *Clam* to be transferred to the higher-order setting without significant modifications. This is important because *Clam* constitutes a significant knowledge base for (inductive) proof. The use of λProlog gives us a powerful meta-theory which allows methods to be written concisely and declaratively, further enhancing one of the advantages of proof planning which is that it produces plans which are understandable to the human user.

As well as being able to prove higher-order theorems which *Clam* cannot, λ*Clam* is also capable of proving many first-order theorems.

Human Reasoning with Diagrams

Keith Stenning
Human Communication Research Centre
Edinburgh University
k.stenning@ed.ac.uk

This talk will review some of our program of research over the past few years into the human use of diagrams. Our aim has been to provide a 'vertically integrated theory' which runs from the semantics of diagrammatic representations, to predicting the details of human performance in diagrammatic tasks. Our claim is that such vertical integration is an absolute requirement for understanding human diagram use. We hope it may provide a point of contact with those whose focus is on automated reasoning.

Diagrams are, for the most part, rather inexpressive in the technical sense of complexity theory, and they therefore lead to tractable reasoning. This point of view has been developed by many authors, notably Levesque [Lev88]. We have taken this observation as a departure for understanding how diagrams aid human inference. Part of that program is to understand what it is in the nature of the semantics of diagrams which leads to inexpressiveness. This in turn has lead to distinguishing two kinds of diagrams, one of which has a 'direct' semantics, and another which has an abstract semantics. Theories of human usability need to treat these two classes of diagrams differently.

There is a paradox in trying to explain modality (diagram vs. sentential language) differences in terms of complexity theory. By hypothesis in complexity theory, inexpressive diagrammatic systems have inexpressive sentential equivalents, so how can one explain modality differences by this route? We attempt to resolve this paradox in terms of the different kinds of access to meta-logical properties that the two kinds of system allow.

We have explored this theoretical approach to the fundamentals in two main empirical areas—software engineering and the teaching of logic. Gurr [Gur95] studied an industrial case study of algebraic and diagrammatic specification of safety critical systems and showed that the two approaches reveal complementary kinds of errors.

Diagrammatic supports for the teaching of logic have a long history. Our comparative studies of diagrammatic and sentential methods show that there are dramatic differences between students responses to the two teaching modalities. The structured nature of teh domain enables deep analysis of the processes that give rise to these different outcomes. Skill in the strategic use of diagrammatic and sentential representations turns out to be the critical factor determining learning success. Having a theory based on complexity of inference allowed detailed predictions of the factors which would affect the success of the diagrammatic methods. Stenning [Ste99] reviews some of this work.

The talk will end with some reflections on the possible relevance of human reasoning to automated reasoning.

References

[Gur95] Gurr, C. A.(1995) Supporting formal reasoning for safety critical systems. *High Integrity Systems*, **1(4)**, pps. 385–396. Oxford University Press.

[Lev88] Levesque, H. (1988) *Logic and the complexity of reasoning. Journal of Philosophical Logic*, **17**, pps. 355–389.

[Ste99] Stenning, K. (1999) Learning to model language and communication. In Brna, P., Baker, M. & Stenning, K. (eds.) Proceedings of the TMR Sponsored Workshop on Learning to Model. Corsica, April 1999. This paper can be ftp-ed as 'stenning.uu' from ~ftp/pubs/graphics on the machine cogsci.ed.ac.uk.

Reasoning with Individuals in Description Logics with Transitive Roles: Preliminary Report

Sergio Tessaris
The University of Manchester, Department of Computer Science
Manchester, M13 9PL, UK
e-mail: `tessaris@cs.man.ac.uk`
WWW: `http://www.man.cs.ac.uk/~tessaris/`

This abstract presents ongoing research on the development of a reasoner for a knowledge representation system based on a Description Logic with transitive roles and individuals. Since Description Logics are syntactic variants of modal logics, the reasoner can be used for checking the satisfiability of multi–modal **S4** formulae with nominals.

Description Logics (DLs) constitute a family of formalisms for the structured representation of information. They developed from research on semantic networks and frames, with the characteristic of having a formally defined semantics which enables sound and complete reasoning. DLs represent knowledge in terms of objects (individuals) grouped into classes (concepts). In addition they provide structuring mechanisms for characterising the relevant properties of classes as relations (roles) between individuals, as well as inclusion relation between concepts (is–a).

This work is based on the DL $\mathcal{CH}_+$, whose concept expressions are build by combining atomic concepts, roles and transitive role names following the abstract syntax:

$$C \longrightarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid \forall R.C$$

where A is a concept name and R is a (transitive) role name. Semantics of such expressions is given in terms of interpretations consisting of a domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$. The interpretation function maps concept expressions to subsets of the domain and (transitive) role names to (transitive) binary relations over the domain. Interpretation functions should preserve the intuitive meaning of the boolean concept forming operators, as well as satisfying the equation $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$ for the $\forall \cdot . \cdot$ operator. The correspondence with propositional multi–modal logics relies on the observation that the elements of the domain can be seen as possible worlds, roles as accessibility relations, and concept names as propositions: the boolean concept forming operators combine formulae as classical negation and conjunction, while the operator $\forall \cdot . \cdot$ corresponds to the multi–modal box operator.

A $\mathcal{CH}_+$ knowledge base is a finite set of inclusion assertions of the forms $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$; as well as membership assertions of the forms $C(\alpha)$ and $R(\alpha_1, \alpha_2)$ about a set of individual names. The semantics of inclusion assertions is the intuitive one (i.e. $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$); moreover membership assertion semantics is given by extending the interpretation functions in such a way that they map each individual name to a different element of the domain (i.e. $(\alpha_1^{\mathcal{I}}, \alpha_2^{\mathcal{I}}) \in R^{\mathcal{I}}$). From the modal logic viewpoint individual names provide names for different possible worlds: role membership assertions partially describe accessibility relations, while concept membership assertions state which formulae are known to be true in the named worlds. An interpretation is said to be a model for a knowledge base if it satisfies each assertion, while a knowledge base is satisfiable if there exists at least one model for it. An efficient knowledge base satisfiability checking algorithm is the basis for a knowledge representation system, because most of the reasoning tasks this kind of system provides can be reduced to a satisfiability check (so called terminological reasoning).

The KB satisfiability algorithm proposed in this work is based on an extension of the so–called precompletion technique [Hol94] for reducing the KB satisfiability problem to concept satisfiability (i.e. satisfiability of propositional multi–modal formulae without nominals). The main idea behind this technique consists in a correctness–preserving process which eliminates dependencies between individuals (induced by $R(\alpha_1, \alpha_2)$ assertions). Once these dependencies are eliminated, the assertions about a single individual can be independently verified (i.e. each named world can be individually considered) ignoring the fact that an individual is involved. The precompletion, or elimination of the dependencies, is performed by adding new assertions using a set of nondeterministic syntactic rules. Because of the nondeterminism of the rules, (exponentially) many different precompletions can be derived from a single knowledge base, which is satisfiable if and only if at least one of these precompletions is satisfiable. Lack of space allows only a brief sketching of the correctness and completeness proof for the technique. The proof is divided in two parts: in the first one it is shown that a knowledge base is satisfiable if and only if at least one of its precompletions is satisfiable, while in the second part is proved that checking the satisfiability of a precompletion can be performed by a sequence of concept satisfiability checks.

The main advantage of this technique lies in the fact that the concept satisfiability tests can be performed by using any available modal checker for **S4**; in particular good result have been recently obtained by optimised tableaux–based systems like FaCT (see [FGM+98]). A first implementation of the algorithm is under development; this implementation will also be used as a testbed for further optimisations which are currently being investigated. Some of the optimisations used for terminological reasoners seems to fit in this framework; as well as the idea that the graph induced by role assertions, can be exploited for driving the precompletion.

The major result of this work consists in proposing a technique which provides hybrid reasoning for a DL with transitive relations. This is a new result in the DL community since the only alternative approach proven to be correct and complete consists in the so–called encoding technique (see [DGL96]) which, till now, has not lead to any practical implementation.

References

[DGL96] Giuseppe De Giacomo and Maurizio Lenzerini. Tbox and abox reasoning in expressive description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann Publishers, 1996.

[FGM+98] Enrico Franconi, Giuseppe De Giacomo, Robert M. MacGregor, Werner Nutt, and Christopher A. Welty, editors. *Proceeding of the 1998 International Workshop on Description Logics (DL'98).* CEUR Pubblication, 1998.

[Hol94] Bernhard Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems.* PhD thesis, Universität des Saarlandes, 1994.

Combining Verification Systems in a Trusted Way to Reap the Benefits of Both

Haiyan Xiong Paul Curzon Ann Blandford
School of Computing Science
Middlesex University, UK

Formal verification techniques can be classified into two categories: deductive theorem proving and symbolic state enumeration. In deductive theorem proving, the verifier states the correctness condition for a system as a theorem in a mathematical logic and a mechanically checked proof of this theorem is generated using a general-purpose theorem prover. In symbolic state enumeration, automated state enumeration techniques provide automation for behavioural comparison or model checking. A tool is used to verify automatically whether or not a description of hardware matches its specification.

In our research, we shall consider two verification systems: the HOL system [3] and the MDG system [1]. The former is an interactive theorem proving system based on higher-order logic, while the latter is an automatic system based on Multiway Decision Graphs. Tahar & Curzon [5] compared these two approaches using each system to verify real hardware: the Fairisle 4 by 4 switch fabric. Their results indicate that each method has complementary advantages and disadvantages. In HOL, the specification language is higher-order logic. The logic is "higher-order" because it allows functions and relations to be passed as arguments to other functions and relations. It also allows a hierarchical verification methodology to be used that can effectively deal with the overall functionality of designs having complex datapaths. To complete a verification, a very detailed understanding of the internal structure is required. However, it is time consuming and leads to difficulties for users when they model and verify a system. For MDG, we have no such trouble because we are not necessarily concerned with the internal structure of the system being verified. The verification procedure is much quicker and more automatic. However, it does not scale to more complex designs.

In the 1990s, the efficiency breakthrough in symbolic state enumeration is such that, industries have successfully applied symbolic state enumeration tools in digital circuit synthesis and verification. More and more companies need experts to help them to verify their products. However, these tools have their restrictions. Can we combine both systems to take advantage of them both? Is the verification system itself correct? In this work we shall show the first step of combining both systems, that is importing MDG results into HOL to form HOL theorems. We shall also investigate the verification of aspects of the MDG system and of the importation process using the HOL system. Each of these areas of work is outlined in the subsequent sections.

Importing the MDG results into HOL. To verify a complex design, we plan to divide it into sub-designs. We shall use MDG to verify the sub-designs and then import the results into HOL. Finally, the design will be proved by HOL. Therefore, the question of how to import the theorem from one system to another system is very important. Generally, in the MDG system, a set of verification applications has been developed including [1]: combinational verification, safety properties checking, sequential verification and counter-example generation. The results of the MDG proof that we call "MDG correctness theorems" have different forms with different verification applications and each of them can be formalised in HOL. However, they are not the theorems that HOL needs if we are to use traditional HOL hardware verification techniques. We shall import the MDG results into the HOL system in a way that they can be used. In other words, we shall prove a series of theorems for different MDG verification applications, and prove from them correctness theorem of the traditional HOL form, i.e. that the structural specification implements the behavioural specification. The translation theorem we prove in each case thus has the form:

$$\text{MDG correctness theorems} \supset$$
$$\text{Structure\_specification} \supset \text{behaviour\_specification}$$

We must convert each MDG "theorem" using a theorem of this form to import it into HOL.

Verified compilation in MDG. If the correctness results of MDG are to be used in a HOL proof, they must be trusted to the same extent as theorems proved in HOL itself. One way to achieve this is to verify aspects of the MDG system in HOL, MDG specifications are written in MDG-HDL. Therefore the first step is to verify the correctness of its implementation. MDG-HDL is a Prolog-style hardware description language, which allows the use of abstract variables for representing data signals.

Some MDG-HDL components are compiled directly into decision graphs. Some MDG-HDL components are compiled into a tabular representation that is a truth table, and then compiled into decision graphs.

We consider the system as a series of translators, translating between different intermediate languages, as shown in Figure 1. First, we give the syntax of the MDG-HDL language, TABLE language and MDG structure language. We then define a set of functions translating MDG-HDL program into the TABLE code and translating the TABLE code into the MDG structure. The correctness of a translator between two languages can be stated in terms of the semantics of the languages. Essentially this states that the translation should preserve the semantics of the source language. This is the traditional form of specification correctness used in the verification of a compiler [4]. We use the same method to specify and verify the MDG system. Curzon, Tahar & Mohamed [2] have formal verified the component library of the MDG system. However, they were not concerned with the syntax. This is necessary if we are to consider the correctness of a translator specification. For the translation to tables, the correctness theorem may have the form,

$$\vdash\ \forall\, \text{p.SemMdgHdl p} = \text{SemMdgTable(TransGT p)}$$

For the translation to an MDG structure the correctness theorem is of the form,

$$\vdash\ \forall\, \text{p.SemMdgTable(TransGT p)} = \text{SemMdgTree(TransTT (TransGT p))}$$

Where p is an MDG-HDL program, SemMdgHdl is the semantics of the source program, SemMdgTable is the semantics of the table subset and TransGT is a functional specification

$$\begin{tikzcd}
\begin{array}{c}\text{MDG-HDL}\\ \text{Syntax}\\ \text{(p)}\end{array} \arrow[r, "\text{(TransGT)}"] \arrow[d, "\text{Semantics}"'] & \begin{array}{c}\text{TABLE}\\ \text{Syntax}\\ \text{(TransGT p)}\end{array} \arrow[r, "\text{(TransTT)}"] \arrow[d, "\text{Semantics}"'] & \begin{array}{c}\text{MDG structure}\\ \text{Syntax}\\ \text{(TransTT(TransGT p))}\end{array} \arrow[d, "\text{Semantics}"'] \\
\begin{array}{c}\text{MDG-HDL}\\ \text{(SemMdgHdl)}\end{array} \arrow[r, leftrightarrow] & \begin{array}{c}\text{TABLE}\\ \text{(SemMdgTable)}\end{array} \arrow[r, leftrightarrow] & \begin{array}{c}\text{MDG structure}\\ \text{(SemMdgTree)}\end{array}
\end{tikzcd}$$

Figure 4: Compilation Correctness

of the translation between the MDG-HDL and TABLE. Similarly SemMdgTree is the semantics of the MDG structure, TransTT is a functional specification of the translation between TABLE and MDG structure.

This work is still on going. To date we have verified that the first translation step is correct for a subset of the MDG-HDL language.

Acknowledgement: This work is funded by a studentship from the School of Computing Science, Middlesex University and by EESRC grant GR/M45221.

References

[1] F. Corella, Z. Zhou, X.Song, M.Langevin, and E. Cerny. Multiway Decision Graphs for Automated Hardware Verification. *Formal Methods in System Design*, 10(1):7-46, 1997.

[2] Paul Curzon, Sofiène Tahar and Otmane Aït-Mohamed. The Verification of Aspects of a Decision Diagram Based Verification System. In Jim Grundy Malcolm Newey (Eds), *Theorem Proving in Higher Order Logics: Emerging Trends,* 11^{th} *International conference*, Department of Computer Science, The Australian National University, p31-46, 1998.

[3] M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-order Logic.* Cambridge University Press 1993.

[4] Jeffrey J. Joyce. A Verified Compiler for a Verified Microprocessor. Technical Report No. 167 University of Cambridge, Computer Laboratory, March 1989

[5] Sofiène Tahar and Paul Curzon. Comparing HOL and MDG: A Case Study on the Verification of An ATM Switch Fabric. To appear in the *Nordic Journal of Computing.*

AISB Convention 1999

Sixth Automated Reasoning Workshop: Bridging the Gap between Theory and Practice

8-9 April 1999

University of Edinburgh, Edinburgh, Scotland

Organisers:	Rob Arthan, Lemma 1	rda@lemma-one.com
	Alan Bundy, Edinburgh	A.Bundy@ed.ac.uk
	Anthony Cohn, Leeds	agc@scs.leeds.ac.uk
	Michael Fisher, Manchester	M.Fisher@doc.mmu.ac.uk
	Alan Frisch, York (Chair of Org. Committee)	frisch@cs.york.ac.uk
	Ian Gent, Strathclyde	ipg@cs.strath.ac.uk
	A. Ireland, Heriot-Watt	A.Ireland@hw.ac.uk
	M. Kerber, Birmingham (Programme Chair, Editor)	M.Kerber@cs.bham.ac.uk
	H. J. Ohlbach, Kings College	ohlbach@dcs.kcl.ac.uk
	Toby Walsh, Strathclyde	tw@cs.strath.ac.uk

RELEVANT WEB SITES:

Workshop:	http://www.cs.bham.ac.uk/~mmk/AR99/
Workshop Series:	http://www.cs.york.ac.uk/~frisch/AutoReason/
AISB-Convention:	http://www.dai.ed.ac.uk/~geraint/aisb99/