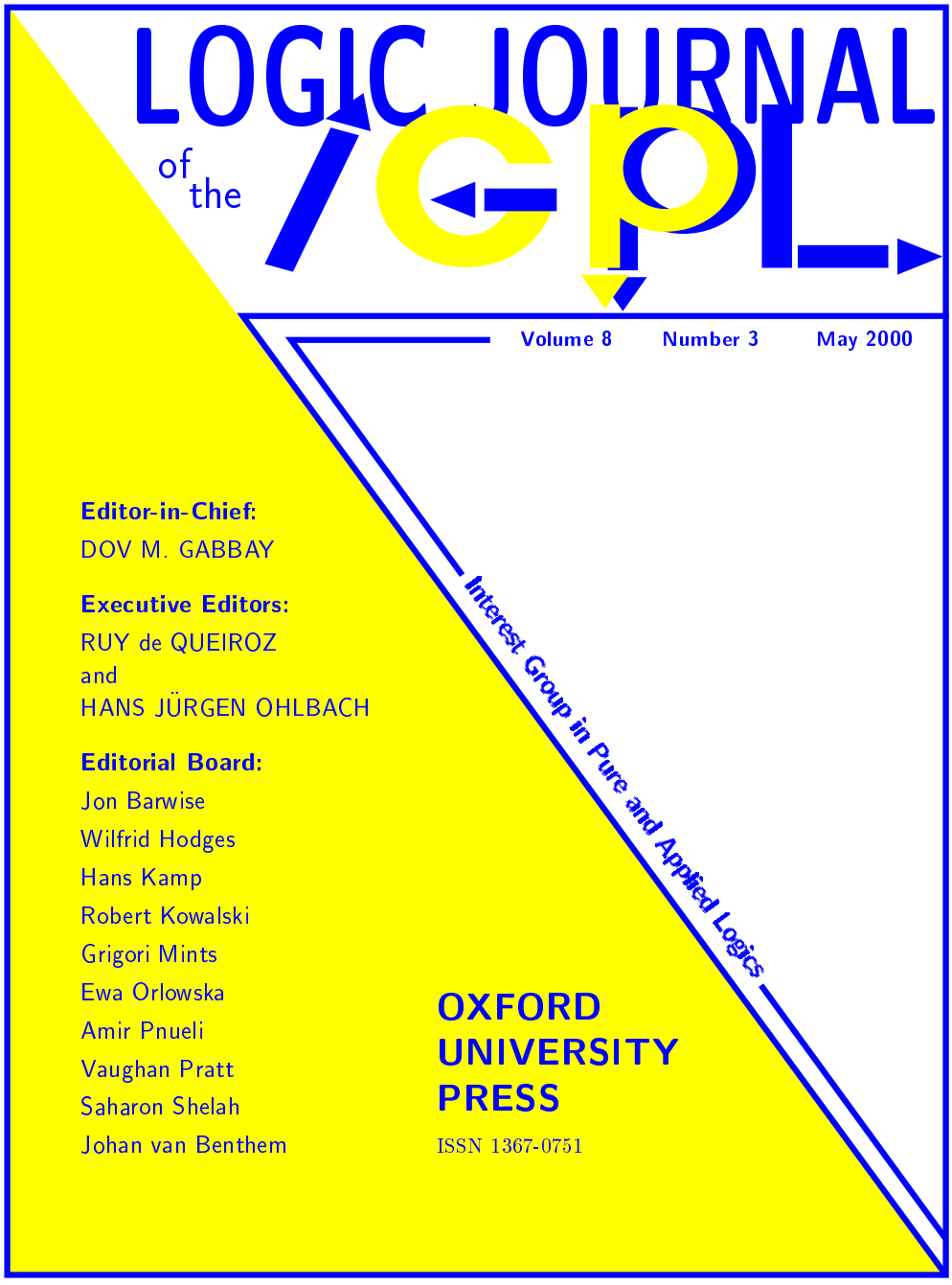

LOGIC JOURNAL
of the
IGPL
Volume 8 Number 3 May 2000
Editor-in-Chief:
DOV M. GABBAY
Executive Editors:
RUY de QUEIROZ
and
HANS JÜRGEN OHLBACH
Editorial Board:
Jon Barwise
Wilfrid Hodges
Hans Kamp
Robert Kowalski
Grigori Mints
Ewa Orlowska
Amir Pnueli
Vaughan Pratt
Saharon Shelah
Johan van Benthem
Interest Group in Pure and Applied Logics
OXFORD
UNIVERSITY
PRESS
ISSN 1367-0751

Subscription Information

Volume 8, 2000 (bimonthly) Full: Europe pounds sterling 275; Rest of World US$ 450. Personal: pounds sterling 138 (US$ 225). Please note that personal rates apply only when copies are sent to a private address and payment is made by a personal cheque or credit card.

Order information

Subscriptions can be accepted for complete volumes only. Prices include air-speeded delivery to Australia, Canada, India, Japan, New Zealand, and the USA. Delivery elsewhere is by surface post. Payment is required with all orders and may be made in the following ways:

Cheque (made payable to Oxford University Press)
National Girobank (account 500 1056)
Credit card (Access, Visa, American Express, Diners Club)
UNESCO Coupons
Bankers: Barclays Bank plc, PO Box 333, Oxford, UK. Code 20-65-18, Account 00715654.

Requests for sample copies, subscription enquiries, orders and changes of address should be sent to the Journals Subscriptions Department, Oxford University Press, Great Clarendon Street, Oxford OX2 6DP, UK. Tel: 01865 267907. Fax: 01865 267485.

Advertisements

Advertising enquiries should be addressed to Peter Carpenter, PRC Associates, The Annexe, Fitznells Manor, Chessington Road, Ewell Village, Surrey KT17 1TF, UK. Tel: 0181 786 7376. Fax: 0181 786 7262.

Copyright

Logic Journal of the IGPL (ISSN 1367-0751) is published bimonthly in January, March, May, July, September and November by Oxford University Press, Oxford, UK. Annual subscription price is US$ 450.00. *Logic Journal of the IGPL* is distributed by M.A.I.L. America, 2323 Randolph Avenue, Avenel, NJ 07001. Periodical postage paid at Rahway, New Jersey, USA and at additional entry points.
US Postmasters: Send address changes to *Logic Journal of the IGPL*, c/o Mercury International, 365 Blair Road, Avenel, NJ 07001, USA.

Back Issues

The current plus two back volumes are available from Oxford University Press. Previous volumes can be obtained from Dawsons Back Issues, Cannon House, Park Farm Road, Folkestone, Kent CT19 5EE, UK. Tel: +44 (0) 1303 203612. Fax: +44 (0) 1303 203617.

Logic Journal of the IGPL

Volume 8, Number 3, May 2000

Contents

Please visit the journal's World Wide Web site at
http://www.jigpal.oupjournals.org

Logic Journal of the Interest Group in Pure and Applied Logics

Scope of the Journal

The *Journal* is the official publication of the International Interest Group in Pure and Applied Logics (IGPL), which is sponsored by The European Foundation for Logic, Language and Information (FoLLI), and currently has a membership of over a thousand researchers in various aspects of logic (symbolic, computational, mathematical, philosophical, etc.) from all over the world.

The *Journal* is published in hardcopy and in electronic form six times per year. Publication is fully electronic: submission, refereeing, revising, typesetting, checking proofs, and publishing, all is done via electronic mailing and electronic publishing.

Papers are invited in all areas of pure and applied logic, including: pure logical systems, proof theory, model theory, recursion theory, type theory, nonclassical logics, nonmonotonic logic, numerical and uncertainty reasoning, logic and AI, foundations of logic programming, logic and computation, logic and language, and logic engineering.

The *Journal* is an attempt to solve a problem in the logic (in particular, IGPL) community:

- Long delays and large backlogs in publication of papers in current journals.
- Very tight time and page number limits on submission.

Papers in the final form should be in LaTeX. The review process is quick, and is made mainly by other IGPL members.

Normal Submissions

Submissions are made by sending a submission letter to the e-mail address: jigpl@dcs.kcl.ac.uk, giving the title and the abstract of the paper, and informing:

(i) of how to obtain the file electronically, if you have the .dvi or .ps file; or

(ii) that you have put the file (.dvi, .ps or .tex) in the public area ftp.dcs.kcl.ac.uk (137.73.8.10), directory, pub/jigpl/submissions

or, by sending 5 (five) hardcopies of the paper to the Editor-in-Chief.

URL: http://www.jigpal.oupjournals.org

Fast Track Submissions for Young Researchers

We offer almost immediate publication for good papers on the basis of two independent recommendations: submit the paper the usual way, but include two referee reports and a letter of recommendation by your 'mentor', for example:

I Professor X would like to act as a fast track editor and introduce the paper by Dr Y entitled 'Title'. I enclose two referee reports by (give names and affiliations of referees). This is an excellent paper, its results are mathematically correct, it has been revised by the author according to the referees' comments and the paper deserves publication in one of your Journals, to be chosen at your discretion.
Sincerely
Professor X

Editorial

CARLOS ARECES, *ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands. E-mail: carlos@wins.uva.nl*

ENRICO FRANCONI, *IMG, Department of Computer Science, University of Manchester, Oxford Rd., Manchester M13 9PL, UK. E-mail: franconi@cs.man.ac.uk*

RAJEEV GORÉ, *Automated Reasoning Project and Department of Computer Science, Australian National University, Canberra, ACT, 0200, Australia. E-mail: Rajeev.Gore@arp.anu.edu.au*

MAARTEN DE RIJKE, *ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands. E-mail: mdr@wins.uva.nl*

HOLGER SCHLINGLOFF, *TZI-BISS, Universität Bremen, P.O. Box 330440, D-28334 HB, Germany. E-mail: hs@tzi.org*

The first Methods for Modalities workshop (M4M-1) was born late at night in November 1998, with a knock on a door, a small but adequate budget, and the urge to do new things. It grew from the good disposition of invited speakers, the enthusiasm of contributors, and long hours behind the computer answering mails, printing articles, organizing time tables, designing web pages and answering phone calls. The aim was clear: to bring together some of the people, logicians and/or computer scientists who were, in one way or another, computing with modal and modal-like logics such as description logic, hybrid logic, temporal logic, etc.

The connections between logic and computing are wide-spread and varied. Well-known examples of uses of logic in computer science include automated verification [25], databases [2], knowledge representation [8], artificial intelligence [20], formal languages [28], etc. Going in the opposite direction, from computer science to logic, we find extremely fast implementations of model checkers and tableaux-based and resolution-based theorem provers [9], automata-theoretic methods for deciding powerful languages [5], tight connections between the theories of computational and descriptive complexity [24], etc. And this is just a small part of a far bigger development, as logic continues to play an important role in computer science and permeating more and more of its main areas. All signs indicate that computer science and logic have decided to establish a stronghold together and profit from the interchange of ideas. This development has been recognized throughout the community, as is witnessed, for instance, by this year's launch of the *ACM Transactions on Computational Logic* [30], the founding of IFCOLOG, the International Federation on Computational Logic [23], and the first installment of the International Conference on Computational Logic [12].

While the links between computer science and *modal* logic may be viewed as nothing more than specific instances of these developments, there is something special to them. Graphs are the key. Graphs are ubiquitous in computer science: think of transition systems, parse trees, Petri nets, decision diagrams, flow charts, ... It is because of this, that modal languages are so well suited to describe com-

puter science phenomena: Kripke models, the standard semantic structure on which modal languages are interpreted, are nothing but graphs [4]. Of course, graphs, or more generally relational structures, are also the semantic structures of choice for other languages, including first-order logic, but from a computational point of view, modal languages have a *huge* advantage when compared to first-order logic: they are usually decidable.

The first workshop on Method for Modalities focused on the particular relationship between modal formalisms and computer science. While planning M4M-1, we aimed to create an environment in which one could really *learn* from what the speakers coming from different fields would have to tell us. We settled on long, tutorial-like presentations by invited speakers, complemented by contributed papers and an afternoon for system demonstrations.

The invited talks covered some of the main approaches to modern automated reasoning (resolution and tableaux calculi), discussed techniques for test set generation, model checking and labeled deduction. Submitted papers, on the other hand, contributed further, less traditional angles. System presentations included both classical and state of the art modal theorem provers, demonstrated by the developers themselves, which helped to create a genuine "hands-on" atmosphere.

The workshop proved to be a stimulating event that motivated people to share knowledge and expertise. Lecture rooms overflowed as we underestimated the popularity of our late-night-idea. And the ripples created by the pebble we threw back in November 1998 are still moving out from the center. Material related to the workshop is available online at the M4M web site (`http://www.illc.uva.nl/~m4m/`); this should serve as an "entry point" for anyone interested in exploring the field. And, of course, M4M-2 is being planned while we write this editorial.

M4M-1 in this Special Issue

Some of the speakers taking part in the workshop were invited to prepare journal versions of their presentation for publication in this special issue. We also invited some of the speakers to join up to form "teams" and provide us with a broad view of their area of expertise. We are extremely glad that the authors concerned took the opportunity and agreed to team up and collaborate.

After a formal reviewing process, five papers were selected as a fair representation of the material presented at M4M-1. Of those five, three focus on methods and methodologies for traditional modal and modal-like formalisms, while two aim to extend familiar views of such formalisms to more inclusive ones, either in terms of general fragments of classical logics or in terms of sorting and naming mechanisms. Let us briefly introduce each of the papers.

Practical Reasoning for Very Expressive Description Logics by Ian Horrocks, Ulrike Sattler, and Stephan Tobies.

Description logics are a family of languages especially devised for knowledge representation [16]. The connection between description logics and modal logics has a long history. The first results date back to [29]; these were extended in, for example, [15].

In application areas such as knowledge representation, one may need very expressive description logics. For instance, one may need to be able to deal with converse relations, number restrictions, A-Box and T-Box reasoning, transitivity, etc. Even though the worst case complexity of the satisfiability problems for these languages is usually EXPTIME, the known algorithms have a good average case performance.

The paper describes one of the most expressive families of description logics nowadays: $\mathcal{ALC}$ extended with transitive roles, hierarchies and converse. It explores the

differences between transitive roles and taking the transitive closure of roles, with respect to tableau-based decision algorithms. It establishes PSPACE-completeness of the satisfiability problem for $\mathcal{SI}$, while an undecidability result is obtained when unrestricted number quantification is also allowed. Finally, the paper discusses various optimization techniques.

The salient characteristic of the paper is its mixture of theoretical and application driven results, which is almost a hallmark of the literature on description logics.

Resolution-Based Methods for Modal Logics by Hans de Nivelle, Renate Schmidt, and Ullrich Hustadt.

Automated theorem proving for first-order logics is a well developed field with years of history [9]. Attending a conference on the field, like for example CADE, International Conference on Automated Deduction, can be an unforgettable experience, where highly tuned heuristics are discussed and test-beds are crunched down by provers trying to outperform each other.

Now, the standard or relational translation [32] embeds modal logics into first-order logics while preserving satisfiability; hence, it opens the door to modal theorem proving by means of first-order techniques. The power of first-order logic translates into both advantages and disadvantages from the modal point of view. On the one hand, it allows one to explore combinations of logics, as different logics can be *jointly* translated into first-order logic; the resulting theory can then be fed to a first-order prover. But, of course, there is the issue of (un-)decidability.

The latter is the topic of the paper, which focuses primarily on first-order resolution methods and how to turn them into decision methods for modal logics. The authors start by introducing the general resolution framework and then go on to discuss refinements of resolution by means of orders and selection functions which are powerful enough to "tame" the method and transform it into a decision procedure for a variety of modal logics.

To complete the picture, the authors also draw connections between resolution and tableau based modal theorem proving, and comment on simulation results.

An Analysis of Empirical Testing for Modal Decision Procedures by Ian Horrocks, Peter F. Patel-Schneider, and Roberto Sebastiani.

It is generally accepted that the worst case complexity of the satisfiability problem for a modal language is not a fair measure to assess the typical case complexity. Many algorithms deciding the same language might live in the same complexity class but perform completely differently when tested on randomly generated or real life problems. Usually, empirical testing is the only way to assess typical case complexity.

The area of testing in propositional logic is well developed and the easy-hard-easy pattern of propositional satisfiability well studied [6]. But little is know about testing for modal logics, which poses a much more complex challenge. For a start, decidable modal languages usually live in one of three different complexity classes: the very easy ones like **S5** in NP, the classical ones in PSPACE like **K** or **T**, and the hard ones like PDL in EXPTIME; and different tests are needed for each of these classes. In addition, the typical behavior of modal logics in those classes need not be the same.

The paper provides a survey of empirical testing methodologies and analyses the current state of the art, proposing criteria for defining a "good test set." In addition, a new test generation methodology is proposed as a variation of the $3\mathrm{CNF}_{\Box_m}$ algorithm.

Reachability Logic: An Efficient Fragment of Transitive Closure Logic by Natasha Alechina and Neil Immerman.

As we said, classical modal languages can be embedded into first-order logic by

means of the standard translation mentioned above. But certain extended modal languages, such as PDL or CTL* [21], allow the Kleene star operator and hence are beyond first-order logic. FO(TC), first-order logic extended with transitive closure [17], is a well studied logic in the field of descriptive complexity. The aim of the paper is to identify fragments of FO(TC) that are general enough to "cover" modal logics like PDL and CTL* but still restricted enough to admit model checking algorithms of very low (linear) complexity.

In particular, the paper introduces a 'modal' fragment of $FO^2(TC)$, the language FO(TC) with only two variables, where quantifiers are restricted by path descriptions, which is later extended with boolean variables. The model checking algorithm for this fragment, which is called $\mathcal{RL}$, is linear in the size of the formula and the size of the model, but exponential in the number of boolean variables. Interestingly, the boolean variables play an important role. The authors show that neither PDL nor CTL* can be embedded in $\mathcal{RL}$ without the use of boolean variables. Furthermore, the number of boolean variables are a good indicator of the complexity of a given CTL* query.

Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto by Patrick Blackburn.

This paper views modal logics as languages that are especially tailored for describing relational structures. After pointing out the connections between modal logics and other fields, it describes a natural extension to traditional modal languages: hybrid logics.

Hybrid logics are languages with the ability to explicitly refer to states in a model. This capacity, which is absent in traditional modal languages, makes hybrid languages especially well-suited for many modeling tasks, including knowledge representation, the analysis of linguistic phenomena, and temporal reasoning. Blackburn starts by discussing simple hybrid logics where only names and satisfaction operators are added. But he also gives a taste of more expressive languages where names can be bound globally or locally, new sorts are introduced, etc.

Blackburn manages to get many important intuitions across in a very accessible manner. Examples abound in the paper, which will help the reader to pull together the many threads from logic, linguistics and computer science that are present in the paper.

M4M-1 not in this Special Issue

For a variety of reasons there is a number of very interesting topics that were covered at M4M-1 but that did not make it into this special issue. In this section we briefly comment on each of them.

Logic offers the possibility of modeling and reasoning about hardware and software systems. But which logic? In his presentation, Basin [3] proposed monadic logics of strings and trees as good candidates for many kinds of discrete systems. The connection between such logics and modal logics is established at the level of frames, and the decidability result of Rabin [27] for SnS has often been used to prove decidability of modal systems by embeddings.

The use of modal and modal-like languages as modeling tools was illustrated in two presentations. Bleeker and Meertens [7] reported on work dealing with modal logics able to capture the dynamics of knowledge during communication, with a view to understanding security protocols. Van Eijck, de Boer, van der Hoek and Meyer [33] presented work on a modal logic with a special kind of quantification, aimed at modeling network topologies.

Traditional proof-theoretical concerns were also represented at M4M-1, where Governatori and Rotolo [22] discussed their recent work on modal proof theory in

the spirit of Gabbay's labeled deductive systems, Fariñas del Cerro and Gasquet [14] presented tableaux-based decision procedures for modal logics of confluence and density, and Ohlbach [26] introduced a new theory resolution style calculus for combined A-Box and T-Box reasoning.

Finally, Cerrito, Mayer and Praud [13] presented work on first-order linear time temporal logics over finite frames; in particular, they discussed a number of results on undecidability and high undecidability.

The following systems were demonstrated at M4M-1: □-KE, developed by Cunningham, Pitt, Williams and Kamara; Akka, developed by Hendriks; FaCT and iFaCT developed by Horrocks; lc2 developed by Marx and Schlobach; Bliksem, developed by de Nivelle; and DLP, developed by Patel-Schneider.

While the scope of M4M-1 was broad and while we managed to attract contributions on a wide variety of topics, it was only a two-day event: various important topics on the interface of modal logic and computing were not addressed during M4M-1, thus suggesting obvious topics for M4M-2 and other future installments of the workshop.

As regards the application of modal logic to hard-core computer science, the work on system verification (like temporal languages for real time systems or the recent advances in model checking [11]) was barely present. Logic programming, and its modal extension to knowledge programming [18], is also a clear topic for M4M, as is, more generally, the connection between modal logics and databases, both at the modeling and inference level. As far as decision methods for modal logics is concerned, there were two important absentees at M4M-1: sequent calculi [34] and automata-based techniques [31]. As we mentioned, modal languages are used as modeling tools in very diverse areas, including computational linguistics, information retrieval, natural language semantics, system design, ... — it would be good to have a fair amount of case studies from as large a subset of these disciplines as possible. Finally, the issue of transfer results for combinations of logics deserves attention [19], as do new connections linking modal languages to game theory [10].

As we tried to convey with the title of this editorial introduction, the message we want to get across is *Use Your Logic*. And we mean *use* in a very concrete way. Modal logics provide restricted yet expressive languages for modeling a wide variety of problems, and today we have automated tools that can perform modal inference with ever increasing efficiency. "Make your life easier, have a theorem prover installed," might be the slogan if an advertising agency were behind M4M. But the phrase is not a mere slogan for us.

Program Committee

The program committee for M4M-1 consisted of Carlos Areces (Amsterdam), Enrico Franconi (Manchester), Rajeev Goré (Canberra), Hans de Nivelle (Saarbrücken), Hans Jürgen Ohlbach (London), Maarten de Rijke (Amsterdam), as well as Holger Schlingloff (Bremen).

Out of 20 submissions, the program committee selected 9 papers for presentation at the workshop. In addition, David Basin, Patrick Blackburn, Ian Horrocks, Hans de Nivelle, Renate Schmidt, and Roberto Sebastiani were invited to address the meeting.

Referees

The program committee gratefully acknowledges the help of the following referees: Guiseppe De Giacomo, Chiara Ghidini, Christoph Lüth, Fabio Massacci, Stephan

Merz, Till Mossakowsk, Markus Roggenbach, George Russell, and Heinrich Wansing.

Sponsors

M4M-1 was sponsored by the Computational Logic group at the Faculty of Science of the University of Amsterdam; the Institute for Logic, Language and Computation (ILLC); Chinaski WorldWide; and the Netherlands Organization for Scientific Research (NWO). We gratefully acknowledge their generous support.

Acknowledgments

We would like to thank Rafael Accorsi, Rosella Gennari, and Marco de Vries for practical assistance before and during M4M-1. We are also grateful to the editors of the *Logic Journal of the IGPL* for their enthusiasm and guidance.

References

[1] C. Areces and M. de Rijke, editors. *Workshop Proceedings Methods for Modalities 1*, Amsterdam, May 1999. ILLC, University of Amsterdam.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, MA, 1995.

[3] D. Basin. Verification based on monadic logic. In Areces and de Rijke [1].

[4] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2000. To appear.

[5] D. Basin and N. Klarlund. Automata based symbolic reasoning in hardware verification. *Formal Methods in Systems Design*, 13:255–288, 1998.

[6] H. Kleine Büning and T. Lettman. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999. Originally published in German by B. G. Teubner, 1994.

[7] A. Bleeker and L. Meertens. Knowledge in security protocols: an operational semantics for BAN logic. In Areces and de Rijke [1].

[8] G. Brewka, editor. *Principles of Knowledge Representation*. CSLI Publications, 1996.

[9] W. Bibel and P. Schmitt, editors. *Automated Deduction – A Basis for Applications*. Kluwer Academic Publishers, 1998.

[10] G. Bonanno and W. van der Hoek. *LOFT-3*. Special issue of *Games and Economic Theory*, 2000.

[11] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[12] CL2000: First International Conference on Computational Logic. URL: `http://www.doc.ic.ac.uk/cl2000/`.

[13] S. Cerrito, M. Mayer, and S. Praud. First-order linear time temporal logic over finite time frames is not semi-decidable. In Areces and de Rijke [1].

[14] L. Fariñas del Cerro and O. Gasquet. Tableaux based decision procedures for modal logics of confluence and density. In Areces and de Rijke [1].

[15] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. 12th National Conference on Artificial Intelligence (AAAI'94)*, pages 205–212, 1994.

[16] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford, 1996.

[17] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.

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

[19] D. Gabbay and M. de Rijke, editors. *Frontiers of Combining Systems 2*. Research Studies Press/Wiley, 2000.

[20] D. Gabbay, C. Hogger, and J. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1. Oxford University Press, Oxford, 1993.

[21] R. Goldblatt. *Logics of Time and Computation*. CSLI Publications, 1992.

[22] G. Governatori and A. Rotolo. Labelled modal sequents. In Areces and de Rijke [1].

[23] IFCOLOG: International Federation for Computational Logic. URL: `http://www.compulog.org/net/IFCoLog.html`.

[24] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[25] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems. Vol. 1, Specification*. Springer-Verlag, New York, 1992.

[26] H. Ohlbach. A theory resolution style A-Box calculus. In Areces and de Rijke [1].

[27] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[28] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*. Springer, 1997.

[29] K. Schild. A correspondence theory for terminological logics. In *Proc. of the 12th IJCAI*, pages 466–471, 1991.

[30] ACM Transactions on Computational Logic. URL: `http://www.acm.org/tocl/`.

[31] M. Vardi. Why is modal logic so robustly decidable? In *DIMACS Ser. Disc. Math. Theoret. Comp. Sci. 31*, pages 149–184. AMS, 1997.

[32] J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. Vol. II*, volume 165 of *Synthese Library*, pages 167–247. D. Reidel Publishing Co., Dordrecht, 1984.

[33] R. van Eijk, F. de Boer, W. van der Hoek, and J. Meyer. A modal interpretation of quantification. In Areces and de Rijke [1].

[34] H. Wansing. *Displaying Modal Logic*. Kluwer, 1998.

Received February 29, 2000

Practical Reasoning for Very Expressive Description Logics

IAN HORROCKS, *Department of Computer Science, University of Manchester, Manchester, UK. E-mail: horrocks@cs.man.ac.uk*

ULRIKE SATTLER, *LuFG Theoretical Computer Science, RWTH Aachen, Germany. E-mail: sattler@informatik.rwth-aachen.de*

STEPHAN TOBIES, *LuFG Theoretical Computer Science, RWTH Aachen, Germany. E-mail: tobies@informatik.rwth-aachen.de*

Abstract

Description Logics (DLs) are a family of knowledge representation formalisms mainly characterised by constructors to build complex concepts and roles from atomic ones. Expressive role constructors are important in many applications, but can be computationally problematical.

We present an algorithm that decides satisfiability of the DL $\mathcal{ALC}$ extended with transitive and inverse roles and functional restrictions with respect to general concept inclusion axioms and role hierarchies; early experiments indicate that this algorithm is well-suited for implementation. Additionally, we show that $\mathcal{ALC}$ extended with just transitive and inverse roles is still in PSPACE. We investigate the limits of decidability for this family of DLs, showing that relaxing the constraints placed on the kinds of roles used in number restrictions leads to the undecidability of all inference problems. Finally, we describe a number of optimisation techniques that are crucial in obtaining implementations of the decision procedures, which, despite the hight worst-case complexity of the problem, exhibit good performance with real-life problems.

Keywords: description logic, modal logic, automated reasoning, tableaux algorithm

1 Motivation

Description Logics (DLs) are a well-known family of knowledge representation formalisms [17]. They are based on the notion of concepts (unary predicates, classes) and roles (binary relations), and are mainly characterised by constructors that allow complex concepts and roles to be built from atomic ones. Sound and complete algorithms for the interesting inference problems such as subsumption and satisfiability of concepts are known for a wide variety of DLs.

Transitive and inverse roles play an important role not only in the adequate representation of complex, aggregated objects [35], but also for reasoning with conceptual data models [9]. Moreover, defining concepts using general concept inclusion axioms seems natural and is crucial for representing conceptual data models.

The relevant inference problems for (an extension of) $\mathcal{ALC}$ augmented in the described manner are known to be decidable [15], and worst-case optimal inference algorithms have been described [16]. However, to the best of our knowledge, nobody has found efficient means to deal with their high degree of non-determinism, which so far prohibits their use in realistic applications. This is mainly due to the fact that

these algorithms can handle not only transitive roles but also the transitive closure of roles. It has been shown [43] that restricting the DL to transitive roles can lead to a lower complexity, and that transitive roles, even when combined with role hierarchies, allow for algorithms that behave quite well in realistic applications [31]. However, until now it has been unclear if this is still true when inverse roles are also present.

In this paper we present various aspects of our research in this direction. Firstly, we motivate our use of logics with transitive roles instead of transitive closure by contrasting algorithms for several pairs of logics that differ only in the kind of transitivity supported.

Secondly, we present an algorithm that decides satisfiability of $\mathcal{ALC}$ extended with transitive and inverse roles, role hierarchies, and functional restrictions. This algorithm can also be used for checking satisfiability and subsumption with respect to general concept inclusion axioms (and thus cyclic terminologies) because these axioms can be "internalised". The fact that our algorithm needs to deal only with transitive roles, instead of transitive closure, leads to a lower degree of non-determinism, and experiments indicate that the algorithm is well-suited for implementation.

Thirdly, we show that $\mathcal{ALC}$ extended with both transitive *and* inverse roles is still in PSPACE. The algorithm used to prove this result introduces an enhanced blocking technique that should also provide useful efficiency gains in implementations of more expressive DLs.

Fourthly, we investigate the limits of decidability for this family of DLs, showing that relaxing the constraints we will impose on the kind of roles allowed in number restrictions leads to the undecidability of all inference problems.

Finally, we describe a range of optimisation techniques that can be used to produce implementations of our algorithms that exhibit good typical case performance.

2 Preliminaries

In this section, we present the syntax and semantics of the various DLs that are investigated in subsequent sections. This includes the definition of inference problems (concept subsumption and satisfiability, and both of these problems with respect to terminologies) and how they are interrelated.

The logics we will discuss are all based on an extension of the well known DL $\mathcal{ALC}$ [45] to include transitively closed primitive roles [43]; we will call this logic $\mathcal{S}$ due to its relationship with the propositional (multi) modal logic $\mathbf{S4_{(m)}}$ [44].[1] This basic DL is then extended in a variety of ways—see Figure 1 for an overview.

Definition 2.1 *Let N_C be a set of* concept names *and* $\mathbf{R}$ *a set of* role names *with transitive role names $\mathbf{R}_+ \subseteq \mathbf{R}$. The set of $\mathcal{SI}$-*roles *is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. To avoid considering roles such as R^{--}, we define a function* Inv *on roles such that $\mathsf{Inv}(R) = R^-$ if R is a role name, and $\mathsf{Inv}(R) = S$ if $R = S^-$. In the following, when speaking of roles, we refer to $\mathcal{SI}$-roles, as our approach is capable of dealing uniformly with both role names and inverse roles.*

Obviously, a role R is transitive iff $\mathsf{Inv}(R)$ is transitive. We therefore define Trans

[1] This logic has previously been called $\mathcal{ALC}_{R^+}$, but this becomes too cumbersome when adding letters to represent additional features.

to return true *iff R is a transitive role. More precisely,* $\mathsf{Trans}(R) = \text{true}$ *(and we say that R is transitive) iff* $R \in \mathbf{R}_+$ *or* $\mathsf{Inv}(R) \in \mathbf{R}_+$.

*The set of $\mathcal{SI}$-*concepts *is the smallest set such that*

1. *every concept name is a concept, and,*
2. *if C and D are concepts and R is an $\mathcal{SI}$-role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are also concepts.*

A role inclusion axiom *is of the form $R \sqsubseteq S$, where R and S are two roles, each of which can be inverse. A* role hierarchy *is a finite set of role inclusion axioms, and $\mathcal{SHI}$ is obtained from $\mathcal{SI}$ by allowing, additionally, for a role hierarchy $\mathcal{R}$. The* sub-role relation $\sqsubseteq^*$ *is the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$.*

$\mathcal{SHIQ}$ is obtained from $\mathcal{SHI}$ by allowing, additionally, for qualified number restrictions [26], i.e., for concepts of the form $\leqslant nR.C$ and $\geqslant nR.C$, where R is a simple *role, C is a concept, and $n \in \mathbb{N}$. A role is called* simple *iff it is neither transitive nor has transitive sub-roles. $\mathcal{SHIN}$ is the restriction of $\mathcal{SHIQ}$ allowing only unqualified number restrictions (i.e., concepts of the form $\leqslant nR$ and $\geqslant nR$), while $\mathcal{SHIF}$ represents a further restriction where, instead of arbitrary number restrictions, only* functional restrictions *of the form $\leqslant 1R$ and their negation $\geqslant 2R$ may occur.*

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a set $\Delta^{\mathcal{I}}$, called the* domain *of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for all concepts C, D, roles R, S, and non-negative integers n, the properties in Figure 1 are satisfied, where $\sharp M$ denotes the cardinality of a set M. An interpretation satisfies a role hierarchy $\mathcal{R}$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$; we denote this fact by $\mathcal{I} \models \mathcal{R}$ and say that $\mathcal{I}$ is a model of $\mathcal{R}$.*

A concept C is called satisfiable *with respect to a role hierarchy $\mathcal{R}$ iff there is some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}$ and $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a* model of C *w.r.t. $\mathcal{R}$. A concept D* subsumes *a concept C w.r.t. $\mathcal{R}$ (written $C \sqsubseteq_{\mathcal{R}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{R}$. For an interpretation $\mathcal{I}$, an individual $x \in \Delta^{\mathcal{I}}$ is called an* instance *of a concept C iff $x \in C^{\mathcal{I}}$.*

All DLs considered here are closed under negation, hence subsumption and (un)satisfiability w.r.t. role hierarchies can be reduced to each other: $C \sqsubseteq_{\mathcal{R}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}$, and C is unsatisfiable w.r.t. $\mathcal{R}$ iff $C \sqsubseteq_{\mathcal{R}} A \sqcap \neg A$ for some concept name A.

In [37; 3; 44; 1], the *internalisation* of terminological axioms is introduced, a technique that reduces reasoning with respect to a (possibly cyclic) *terminology* to satisfiability of concepts. In [31], we saw how role hierarchies can be used for this reduction. In the presence of inverse roles, this reduction must be slightly modified.

Definition 2.2 *A* terminology $\mathcal{T}$ *is a finite set of general concept inclusion axioms, $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, where C_i, D_i are arbitrary $\mathcal{SHIF}$-concepts. An interpretation $\mathcal{I}$ is said to be a* model *of $\mathcal{T}$ iff $C_i^{\mathcal{I}} \subseteq D_i^{\mathcal{I}}$ holds for all $C_i \sqsubseteq D_i \in \mathcal{T}$. A concept C is* satisfiable *with respect to $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Finally, D* subsumes *C with respect to $\mathcal{T}$ iff, for each model $\mathcal{I}$ of $\mathcal{T}$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.*

The following lemma shows how general concept inclusion axioms can be *internalised* using a "universal" role U, a transitive super-role of all roles occurring in $\mathcal{T}$ and their respective inverses.

Construct Name	Syntax	Semantics	
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$\mathcal{S}$
atomic role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	
transitive role	$R \in \mathbf{R}_+$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
exists restriction	$\exists R.C$	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$	
value restriction	$\forall R.C$	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$	
role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{H}$
inverse role	R^-	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$	$\mathcal{I}$
number restrictions	$\geqslant nR$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\}$	$\mathcal{N}$
	$\leqslant nR$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\}$	
qualifying number restrictions	$\geqslant nR.C$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\}$	$\mathcal{Q}$
	$\leqslant nR.C$	$\{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\}$	

FIG. 1. Syntax and semantics of the $\mathcal{SI}$ family of DLs

Lemma 2.3 *Let $\mathcal{T}$ be a terminology, $\mathcal{R}$ a role hierarchy, and C, D $\mathcal{SHIF}$-concepts, and let*

$$C_{\mathcal{T}} := \bigsqcap_{C_i \sqsubseteq D_i \in \mathcal{T}} \neg C_i \sqcup D_i.$$

Let U be a transitive role that does not occur in $\mathcal{T}, C, D,$ or $\mathcal{R}$. We set

$$\mathcal{R}_U := \mathcal{R} \cup \{R \sqsubseteq U, \mathsf{Inv}(R) \sqsubseteq U \mid R \text{ occurs in } \mathcal{T}, C, D, \text{ or } \mathcal{R}\}.$$

Then C is satisfiable w.r.t. $\mathcal{T}$ and $\mathcal{R}$ iff $C \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is satisfiable w.r.t. $\mathcal{R}_U$. Moreover, D subsumes C w.r.t. $\mathcal{T}$ and $\mathcal{R}$ iff $C \sqcap \neg D \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is unsatisfiable w.r.t. $\mathcal{R}_U$.

The proof of Lemma 2.3 is similar to the ones that can be found in [44; 3]. Most importantly, it must be shown that, (a) if a $\mathcal{SHIF}$-concept C is satisfiable with respect to a terminology $\mathcal{T}$ and a role hierarchy $\mathcal{R}$, then C, $\mathcal{T}$, and $\mathcal{R}$ have a *connected* model, and (b) if y is reachable from x via a role path (possibly involving inverse roles) in a model of $\mathcal{T}$ and $\mathcal{R}_U$, then $\langle x, y\rangle \in U^{\mathcal{I}}$. These are easy consequences of the semantics and the definition of U.

Theorem 2.4 *Satisfiability and subsumption of $\mathcal{SHIF}$-concepts (resp. $\mathcal{SHI}$-concepts) w.r.t. terminologies and role hierarchies are polynomially reducible to (un)satisfiability of $\mathcal{SHIF}$-concepts (resp. $\mathcal{SHI}$-concepts) w.r.t. role hierarchies.*

3 Blocking

The algorithms we are going to present for deciding satisfiability of $\mathcal{SI}$- and $\mathcal{SHIF}$-concepts use the tableaux method [25], in which the satisfiability of a concept D is

tested by trying to construct a model of D. The model is represented by a tree in which nodes correspond to individuals and edges correspond to roles. Each node x is labelled with a set of concepts $\mathfrak{L}(x)$ that the individual x must satisfy, and edges are labelled with (sets of) role names.

An algorithm starts with a single node labelled $\{D\}$, and proceeds by repeatedly applying a set of *expansion rules* that recursively decompose the concepts in node labels, new edges and nodes being added as required in order to satisfy $\exists R.C$ or $(\geqslant 2\ F)$ concepts. The construction terminates either when none of the rules can be applied in a way that extends the tree, or when the discovery of obvious contradictions demonstrates that D has no model.

In order to prove that such an algorithm is a sound and complete decision procedure for concept satisfiability in a given logic, it is necessary to demonstrate that the models it constructs are correct with respect to the semantics, that it will always find a model if one exists, and that it always terminates. The first two points can usually be dealt with by proving that the expansion rules preserve satisfiability, and that in the case of non-deterministic expansion (e.g., of disjunctions) all possibilities are exhaustively searched. For logics such as $\mathcal{ALC}$, termination is mainly due to the fact that the expansion rules can only add new concepts that are strictly smaller than the decomposed concept, so the model must stabilise when all concepts have been fully decomposed. As we will see, this is no longer true in the presence of transitive roles.

3.1 Transitive Roles vs. Transitive Closure

We have argued that reasoning for logics with transitive roles is empirically more tractable than for logics that allow for transitive closure of roles [43; 31]. In this section we will give some justification for that claim. The starting point for our investigations are the logics $\mathcal{SH}$ [31] and $\mathcal{ALC}_+$ [3], which extend $\mathcal{ALC}$ by transitive roles and role hierarchies or transitive closure of roles respectively. Syntactically, $\mathcal{ALC}_+$ is similar to $\mathcal{S}$, where, in addition to transitive and non-transitive roles, the transitive closure R^+ of a role R may appear in existential and universal restrictions. Formally, R^+ is interpreted by

$$(R^+)^{\mathcal{I}} = \bigcup_{i\in\mathbb{N}} (R^{\mathcal{I}})^i, \quad \text{where } (R^{\mathcal{I}})^i = \begin{cases} R^{\mathcal{I}}, & \text{if } i = 1 \\ R^{\mathcal{I}} \circ (R^{\mathcal{I}})^{i-1}, & \text{otherwise} \end{cases}$$

For both $\mathcal{SH}$ and $\mathcal{ALC}_+$, concept satisfiability is an EXPTIME-complete problem. This result is easily derived from the EXPTIME-hardness proof for PDL in [18] and from the proof that PDL is in EXPTIME in [41]. Nevertheless, implementations of algorithms for $\mathcal{SH}$ exhibit good performance in realistic applications [34] whereas, at the moment, this seems to be more problematical for $\mathcal{ALC}_+$. We believe that the main reason for this discrepancy, at least in the case of tableau algorithm implementations, lies in the different complexity of the blocking conditions that are needed to guarantee the termination of the respective algorithms. In the following we are going to survey the blocking techniques needed to deal with $\mathcal{SH}$ and its subsequent extensions to $\mathcal{SHI}$ and $\mathcal{SHIF}$. To underpin our claim that reasoning with transitive roles empirically leads to more efficient implementations than for transitive closure, we will also present the blocking techniques used to deal with transitive closure. These are more complicated

and introduce a larger degree of non-determinism into the tableaux algorithms, leading to inferior performance of implementations.

3.2 Blocking for $\mathcal{S}$ and $\mathcal{SH}$

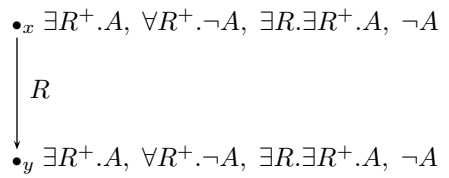
Termination of the expansion process of a tableaux algorithm is not guaranteed for logics that include transitive roles, as the expansion rules can introduce new concepts that are the same size as the decomposed concept. In particular, $\forall R.C$ concepts, where R is a transitive role, are dealt with by propagating the whole concept across R-labelled edges [43]. For example, given a node x labelled $\{C, \exists R.C, \forall R.(\exists R.C)\}$, where R is a transitive role, the combination of the $\exists R.C$ and $\forall R.(\exists R.C)$ concepts would cause a new node y to be added to the tree with a label identical to that of x. The expansion process could then be repeated indefinitely.

This problem can be dealt with by *blocking*: halting the expansion process when a *cycle* is detected [3; 8]. For logics without inverse roles, the general procedure is to check the label of each new node y, and if it is a *subset* [2] of the label of an ancestor node x, then no further expansion of y is performed: x is said to block y. The resulting tree corresponds to a cyclical model in which y is identified with x.

To deal with the transitive closure of roles, tableaux algorithms proceed by non-deterministically expanding a concept $\exists R^+.C$ to either $\exists R.C$ or $\exists R.\exists R^+.C$. Again, since the size of concepts along a path in the tree may not decrease, blocking techniques are necessary to guarantee termination. An adequate blocking condition for $\mathcal{ALC}_+$ is identical as for $\mathcal{SH}$, but one has to distinguish between *good* and *bad* cycles. Consider the following concept:

$$D = \exists R^+.A \sqcap \forall R^+.\neg A \sqcap \neg A$$

While D is obviously not satisfiable, a run of a tableaux algorithm might generate the following tableau in which node y is blocked by node x without generating any obvious contradictions.



The problem is that $\exists R^+.A$ has always been expanded to $\exists R.\exists R^+.A$, postponing the satisfaction of A a further step. To obtain a correct tableaux algorithm for $\mathcal{ALC}_+$, the blocking condition must include a check to ensure that each concept $\exists R^+.C$ appearing in such a cycle is expanded to $\exists R.C$ somewhere in the cycle. Such cycles are called *good* cycles, whereas cycles in which $\exists R^+.C$ has always been expanded to $\exists R.\exists R^+.C$ are called *bad* cycles. A valid model may only contain good cycles.

Summing up, using transitive closure instead of transitive roles has a twofold impact on the empirical tractability: (a) in blocking situations, good cycles have to be distinguished from bad ones, and (b) the non-deterministic expansion of concepts of the form $\exists R^+.C$ increases the size of the search space.

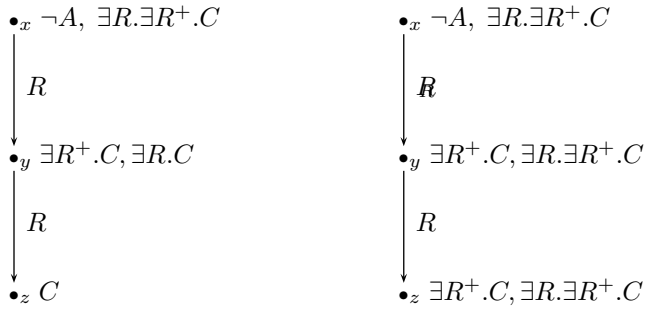


FIG. 2. Dynamic blocking fails in the presence of transitive closure.

3.3 Adding Inverse Roles

Blocking is more problematical when inverse roles are added to the logic, and a key feature of the algorithms presented in [35] was the introduction of a *dynamic blocking* strategy. Besides using label equality instead of subset, this strategy allowed blocks to be established, broken, and re-established. With inverse roles the blocking condition has to be considered more carefully because roles are now bi-directional, and additional concepts in x's label could invalidate the model with respect to y's predecessor. This problem can be overcome by allowing a node x to be blocked by one of its ancestors y if and only if they were labelled with the same sets of concepts.

Dealing with inverse roles is even more complicated in the presence of transitive closure. As an example consider the following concept:

$$D = \neg A \sqcap \exists R.\exists R^+.C$$
$$C = \forall R^-.(\forall R^-.A)$$

Fig. 2 shows two possible tableau expansions of the concept D. Continuing the expansion of the left hand tree will necessarily lead to a clash when concept $C \in \mathcal{L}(z)$ is expanded as this will lead to both A and $\neg A$ appearing in $\mathcal{L}(x)$. The right hand tree is also invalid as it contains a bad cycle: $\mathcal{L}(y) = \mathcal{L}(z)$ but $\exists R^+.D$ has always been expanded to $\exists R.\exists R^+.D$. Nevertheless, D is satisfiable, as it would be shown by continuing the expansion of the right hand path for one more step.

In [16], a solution to this problem for CPDL, a strict superset of $\mathcal{ALCI}_+$ ($\mathcal{ALC}_+$ plus inverse roles) is presented. The solution consists of an additional expansion rule called the *look behind analytical cut*. This rule employs exhaustive non-deterministic guessing to make the past of each node in the tree explicit in the labelling of that node: if y is an R-successor of a node x, then $\exists R^-.C$ or $\forall R^-.\neg C$ is added non-deterministically to the label of y for each concept C that may appear during the expansion process. Obviously, this leads to a further large increase in the size of the search space, with a correspondingly large adverse impact on empirical tractability. Experience with this kind of exhaustive guessing leads us to believe that an implementation of such an algorithm would be disastrously inefficient. The non-existence of implementations for $\mathcal{ALCI}_+$ or CPDL might be taken to support this view.

3.4 Pair-wise Blocking

Further extending the logic $\mathcal{SHI}$ to $\mathcal{SHIF}$ by adding functional restrictions (concepts of the form $(\leqslant 1\ R)$, meaning that an individual can be related to at most one other individual by the role R) introduces new problems associated with the fact that the logic no longer has the finite model property. This means that there are concepts that are satisfiable but for which there exists no finite model. An example of such a concept is

$$\neg C \sqcap \exists F^{-}.(C \sqcap (\leqslant 1\ F)) \sqcap \forall R^{-}.(\exists F^{-}.(C \sqcap (\leqslant 1\ F))),$$

where R is a transitive role and $F \sqsubseteq R$. Any model of this concept must contain an infinite sequence of individuals, each related to a single successors by an F^{-} role, and each satisfying $C \sqcap \exists F^{-}.C$, the $\exists F^{-}.C$ term being propagated along the sequence by the transitive super-role R. Attempting to terminate the sequence in a cycle causes the whole sequence to collapse into a single node due to the functional restrictions $(\leqslant 1\ F)$, and this results in a contradiction as both C and $\neg C$ will be in that node's label.

In order to deal with infinite models—namely to have an algorithm that terminates correctly even if the input concept has only infinite models—a more sophisticated *pair-wise* blocking strategy was introduced in [35], and soundness was proved by demonstrating that a blocked tree always has a corresponding infinite model.[2]

The only known algorithm that is able to deal with the combination of transitive closure, inverse roles, and functional restrictions on roles relies on an elaborate polynomial reduction to a CPDL terminology [13], and the capability of CPDL to internalise the resulting general terminological axioms. The large number and the nature of the axioms generated by this reduction make it very unlikely that an implementation with tolerable runtime behaviour will ever emerge.

4 Reasoning for $\mathcal{SI}$ Logics

In this section, we present two tableaux algorithms: the first decides satisfiability of $\mathcal{SHIF}$-concepts, and can be used for all $\mathcal{SHIF}$ reasoning problems (see Theorem 2.4); the second decides satisfiability (and hence subsumption) of $\mathcal{SI}$-concepts in PSPACE. In this paper we only sketch most of the proofs. For details on the $\mathcal{SHIF}$-algorithm, please refer to [35], for details on the $\mathcal{SI}$- and $\mathcal{SIN}$-algorithm, please refer to [27].

The correctness of the algorithms can be proved by showing that they create a *tableau* for a concept iff it is satisfiable.

For ease of construction, we assume all concepts to be in *negation normal form* (NNF), that is, negation occurs only in front of concept names. Any $\mathcal{SHIF}$-concept can easily be transformed to an equivalent one in NNF by pushing negations inwards [25].

Definition 4.1 *Let D be a $\mathcal{SHIF}$-concept in NNF, $\mathcal{R}$ a role hierarchy, and $\mathbf{R}_D$ the set of roles occurring in D together with their inverses, and $sub(D)$ the subconcepts of D. Then $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a* tableau *for D w.r.t. $\mathcal{R}$ iff $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{sub(D)}$ maps each individual to a set of concepts, $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each role to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that*

[2] This is not to say that it may not also have a finite model.

$D \in \mathfrak{L}(s)$. Furthermore, for all $s,t \in \mathbf{S}$, $C,E \in sub(D)$, and $R,S \in \mathbf{R}_D$, it holds that:

1. *if $C \in \mathfrak{L}(s)$, then $\neg C \notin \mathfrak{L}(s)$,*
2. *if $C \sqcap E \in \mathfrak{L}(s)$, then $C \in \mathfrak{L}(s)$ and $E \in \mathfrak{L}(s)$,*
3. *if $C \sqcup E \in \mathfrak{L}(s)$, then $C \in \mathfrak{L}(s)$ or $E \in \mathfrak{L}(s)$,*
4. *if $\forall R.C \in \mathfrak{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(R)$, then $C \in \mathfrak{L}(t)$,*
5. *if $\exists R.C \in \mathfrak{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s,t\rangle \in \mathcal{E}(R)$ and $C \in \mathfrak{L}(t)$,*
6. *if $\forall S.C \in \mathfrak{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^* S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathfrak{L}(t)$,*
7. *$\langle s,t\rangle \in \mathcal{E}(R)$ iff $\langle t,s\rangle \in \mathcal{E}(\mathsf{Inv}(R))$.*
8. *if $\langle x,y\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* S$, then $\langle x,y\rangle \in \mathcal{E}(S)$,*
9. *if $\leqslant 1 R \in \mathfrak{L}(s)$, then $\sharp\{t \mid \langle s,t\rangle \in \mathcal{E}(R)\} \leq 1$, and*
10. *if $\geqslant 2 R \in \mathfrak{L}(s)$, then $\sharp\{t \mid \langle s,t\rangle \in \mathcal{E}(R)\} \geq 2$.*

Tableaux for $\mathcal{SI}$-concepts are defined analogously and must satisfy Properties 1-7, where, due to the absence of a role hierarchy, $\sqsubseteq^*$ is the identity.

Due to the close relationship between models and tableaux, the following lemma can be easily proved by induction on the structure of concepts. As a consequence, an algorithm that constructs (if possible) a tableau for an input concept is a decision procedure for satisfiability of concepts.

Lemma 4.2 *A $\mathcal{SHIF}$-concept (resp. $\mathcal{SI}$-concept) D is satisfiable w.r.t. a role hierarchy $\mathcal{R}$ iff D has a tableau w.r.t. $\mathcal{R}$.*

4.1 Reasoning in $\mathcal{SHIF}$

In the following, we give an algorithm that, given a $\mathcal{SHIF}$-concept D, decides the existence of a tableaux for D. We implicitly assume an arbitrary but fixed role hierarchy $\mathcal{R}$.

Definition 4.3 *A* completion tree *for a $\mathcal{SHIF}$-concept D is a tree where each node x of the tree is labelled with a set $\mathfrak{L}(x) \subseteq sub(D)$ and each edge $\langle x,y\rangle$ is labelled with a set $\mathfrak{L}(\langle x,y\rangle)$ of (possibly inverse) roles occurring in $sub(D)$.*

Given a completion tree, a node y is called an R-successor *of a node x iff y is a successor of x and $S \in \mathfrak{L}(\langle x,y\rangle)$ for some S with $S \sqsubseteq^* R$. A node y is called an* R-neighbour *of x iff y is an R-successor of x, or if x is an $\mathsf{Inv}(R)$-successor of y. Predecessors and ancestors are defined as usual.*

A node is blocked *iff it is directly or indirectly blocked. A node x is* directly blocked *iff none of its ancestors are blocked, and it has ancestors x', y and y' such that*

1. *x is a successor of x' and y is a successor of y'* and
2. *$\mathfrak{L}(x) = \mathfrak{L}(y)$ and $\mathfrak{L}(x') = \mathfrak{L}(y')$* and
3. *$\mathfrak{L}(\langle x',x\rangle) = \mathfrak{L}(\langle y',y\rangle)$.*

In this case we will say that y blocks x.

A node y is indirectly blocked *iff one of its ancestors is blocked, or—in order to avoid wasted expansion after an application of the $\leqslant$-rule—it is a successor of a node x and $\mathfrak{L}(\langle x,y\rangle) = \emptyset$.*

$\sqcap$-rule: if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$

$\sqcup$-rule: if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
then, for some $C \in \{C_1, C_2\}$, $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$

$\exists$-rule: if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and
2. x has no S-neighbour y with $C \in \mathcal{L}(y)$
then create a new node y with
$\mathcal{L}(\langle x, y\rangle) = \{S\}$ and $\mathcal{L}(y) = \{C\}$

$\forall$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$

$\forall_+$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked,
2. there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, and
3. x has an R-neighbour y with $\forall R.C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall R.C\}$

$\geqslant$-rule: if 1. $(\geqslant 2\ R) \in \mathcal{L}(x)$, x is not blocked, and
2. there is no R-neighbour y of x with $A \in \mathcal{L}(y)$
then create two new nodes y_1, y_2 with
$\mathcal{L}(\langle x, y_1\rangle) = \mathcal{L}(\langle x, y_2\rangle) = \{R\}$,
$\mathcal{L}(y_1) = \{A\}$ and $\mathcal{L}(y_2) = \{\neg A\}$

$\leqslant$-rule: if 1. $(\leqslant 1\ R) \in \mathcal{L}(x)$, x is not indirectly blocked,
2. x has two R-neighbours y and z s.t. y is not an ancestor of z,
then 1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and
2. if z is an ancestor of y
then $\mathcal{L}(\langle z, x\rangle) \longrightarrow \mathcal{L}(\langle z, x\rangle) \cup \mathsf{Inv}(\mathcal{L}(\langle x, y\rangle))$
else $\mathcal{L}(\langle x, z\rangle) \longrightarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$
3. $\mathcal{L}(\langle x, y\rangle) \longrightarrow \emptyset$

FIG. 3. The tableaux expansion rules for $\mathcal{SHIF}$

For a node x, $\mathcal{L}(x)$ is said to contain a clash *iff $\{A, \neg A\} \subseteq \mathcal{L}(x)$ or $\{\geqslant 2R, \leqslant 1S\} \subseteq \mathcal{L}(x)$ for roles $R \sqsubseteq^* S$. A completion tree is called* clash-free *iff none of its nodes contains a clash; it is called* complete *iff none of the expansion rules in Figure 3 is applicable.*

For a $\mathcal{SHIF}$-concept D in NNF, the algorithm starts with a completion tree consisting of a single node x with $\mathcal{L}(x) = \{D\}$. It applies the expansion rules, stopping when a clash occurs, and answers "D is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree.

The soundness and completeness of the tableaux algorithm is an immediate consequence of Lemmas 4.2 and 4.4.

Lemma 4.4 *Let D be an $\mathcal{SHIF}$-concept.*

1. *The tableaux algorithm terminates when started with D.*
2. *If the expansion rules can be applied to D such that they yield a complete and*

clash-free completion tree, then D has a tableau.

3. *If D has a tableau, then the expansion rules can be applied to D such that they yield a complete and clash-free completion tree.*

Before we sketch the ideas of the proof, we will discuss the different expansion rules and their correspondence to the language constructors.

The $\sqcap$-, $\sqcup$-, $\exists$- and $\forall$-rules are the standard $\mathcal{ALC}$ tableaux rules [45]. The $\forall_+$-rule is used to handle transitive roles, where the $\stackrel{*}{\sqsubseteq}$-clause deals with the role hierarchy. See [35] for details.

The functional restriction rules merit closer consideration. In order to guarantee the satisfaction of a $\geqslant 2R$-constraint, the $\geqslant$-rule creates two successors and uses a fresh atomic concept A to prohibit identification of these successors by the $\leqslant$-rule. If a node x has two or more R-neighbours and contains a functional restriction $\leqslant 1R$, then the $\leqslant$-rule merges two of the neighbours *and* also merges the edges connecting them with x. Labelling edges with sets of roles allows a single node to be both an R and S-successor of x even if R and S are not comparable by $\stackrel{*}{\sqsubseteq}$. Finally, contradicting functional restrictions are taken care of by the definition of a clash.

We now sketch the main ideas behind the proof of Lemma 4.4:

1. Termination: Let $m = |sub(D)|$ and $n = |\mathbf{R}_D|$. Termination is a consequence of the following properties of the expansion rules:

(a) The expansion rules never remove nodes from the tree or concepts from node labels. Edge labels can only be changed by the $\leqslant$-rule which either expands them or sets them to $\emptyset$; in the latter case the node below the $\emptyset$-labelled edge is blocked. (b) Successors are only generated for concepts of the form $\exists R.C$ and $\geqslant 2R$. For a node x, each of these concepts triggers the generation of at most two successors. If for one of these successors y the $\leqslant$-rule subsequently causes $\mathcal{L}(\langle x, y\rangle)$ to be changed to $\emptyset$, then x will have some R-neighbour z with $\mathcal{L}(z) \supseteq \mathcal{L}(y)$. This, together with the definition of a clash, implies that the concept that led to the generation of y will not trigger another rule application. Obviously, the out-degree of the tree is bounded by $2m$. (c) Nodes are labelled with non-empty subsets of $sub(D)$ and edges with subsets of $\mathbf{R}_D$, so there are at most 2^{2mn} different possible labellings for a pair of nodes and an edge. Therefore, on a path of length at least 2^{2mn} there must be 2 nodes x, y such that x is directly blocked by y. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^{2mn}.

2. Soundness: A complete and clash-free tree $\mathbf{T}$ for D induces the existence of a tableaux $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D as follows. Individuals in $\mathbf{S}$ correspond to *paths* in $\mathbf{T}$ from the root node to some node that is not blocked. Instead of going to a directly blocked node, these paths jump back to the blocking node, which yields paths of arbitrary length. Thus, if blocking occurs, this construction yields an infinite tableau. This rather complicated tableau construction is necessary due to the presence of functional restrictions; its validity is ensured by the blocking condition, which considers both the blocked node and its predecessor.

3. Completeness: A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D can be used to "steer" the application of the non-deterministic $\sqcup$- and $\leqslant$-rules in a way that yields a complete and clash-free tree.

The following theorem is an immediate consequence of Lemma 4.4, Lemma 4.2, and Lemma 2.3.

Theorem 4.5 *The tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{SHIF}$-concepts with respect to terminologies and role hierachies.*

4.2 A PSPACE*-algorithm for $\mathcal{SI}$*

To obtain a PSPACE-algorithm for $\mathcal{SI}$, the $\mathcal{SHIF}$ algorithm is modified as follows: (a) As $\mathcal{SI}$ does not allow for functional restrictions, the $\geqslant$- and the $\leqslant$-rule can be omitted; blocking no longer involves two pairs of nodes with identical labels but only two nodes with "similar" labels. (b) Due to the absence of role hierarchies, edge labels can be restricted to roles (instead of sets of roles). (c) To obtain a PSPACE algorithm, we employ a refined blocking strategy which necessitates a second label $\mathcal{B}$ for each node. This blocking technique, while discovered independently, is based on ideas similar to those used in [46] to show that satisfiability for $\mathbf{K4_t}$ can be decided in PSPACE.[3] In the following, we will describe and motivate this blocking technique; detailed proofs as well as a similar result for $\mathcal{SIN}$ can be found in [27].

Please note that naively using a cut rule does not yield a PSpace algorithm: a cut rule similar to the *look behind analytical cut* presented in [16] (non-deterministically) guesses which constraints will be propagated "up" the completion tree by universal restrictions on inverted roles. For $\mathcal{SI}$, this technique may lead to paths of exponential length due to equality blocking. A way to avoid these long paths would be to stop the investigation of a path at some polynomial bound. However, to prove the correctness of this approach, it would be necessary to establish a "short-path-model" property similar to Lemma 4.8. Furthermore, we believe that our algorithm is better suited for an implementation since it makes less use of "don't-know" non-determinism. This also distinguishes our approach from the algorithm presented in [46], which is not intended to form the basis for an efficient implementation.

Definition 4.6 *A* completion tree *for a $\mathcal{SI}$ concept D is a tree where each node x of the tree is labelled with two sets $\mathcal{B}(x) \subseteq \mathcal{L}(x) \subseteq sub(D)$ and each edge $\langle x, y\rangle$ is labelled with a (possibly inverse) role $\mathcal{L}(\langle x, y\rangle)$ occurring in $sub(D)$.*

R-neighbours, -successors, and -predecessors are defined as in Definition 4.3. Due to the absence of role hierarchies, $\sqsubseteq^*$ is the identity on $\mathbf{R}_D$.

A node x is blocked *iff, for an ancestor y, y is blocked or*

$$\mathcal{B}(x) \subseteq \mathcal{L}(y) \quad \text{and} \quad \mathcal{L}(x)/\,\mathsf{Inv}(S) = \mathcal{L}(y)/\,\mathsf{Inv}(S),$$

where x' is the predecessor of x, $\mathcal{L}(\langle x', x\rangle) = S$, and $\mathcal{L}(x)/\,\mathsf{Inv}(S) = \{\forall\,\mathsf{Inv}(S).C \in \mathcal{L}(x)\}$.

For a node x, $\mathcal{L}(x)$ is said to contain a clash *iff $\{A, \neg A\} \subseteq \mathcal{L}(x)$. A completion tree to which none of the expansion rules given in Figure 4 is applicable is called* complete.

For an $\mathcal{SI}$-concept D, the algorithm starts with a completion tree consisting of a single node x with $\mathcal{B}(x) = \mathcal{L}(x) = \{D\}$. It applies the expansion rules in Figure 4, stopping when a clash occurs, and answers "D is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree.

As for $\mathcal{SHIF}$, correctness of the algorithm is proved by first showing that a $\mathcal{SI}$-concept is satisfiable iff it has a tableau, and next proving the $\mathcal{SI}$-analogue of Lemma 4.4.

[3] The modal logic $\mathbf{K4_t}$ is a syntactic variant of $\mathcal{SI}$ with only a single transitive role name.

$\sqcap$-rule: if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and
2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$

$\sqcup$-rule: if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and
2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$\forall$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$ and
2. there is an S-successor y of x with $C \notin \mathcal{B}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$ and
$\mathcal{B}(y) \longrightarrow \mathcal{B}(y) \cup \{C\}$ or
2'. there is an S-predecessor y of x with $C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$.

$\forall_+$-rule: if 1. $\forall S.C \in \mathcal{L}(x)$ and $\mathsf{Trans}(S)$ and
2. there is an S-successor y of x with $\forall S.C \notin \mathcal{B}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall S.C\}$ and
$\mathcal{B}(y) \longrightarrow \mathcal{B}(y) \cup \{\forall S.C\}$ or
2'. there is an S-predecessor y of x with $\forall S.C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall S.C\}$.

$\exists$-rule: if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked and no other rule is applicable to any of its ancestors, and
2. x has no S-neighbour y with $C \in \mathcal{L}(y)$
then create a new node y with
$\mathcal{L}(\langle x, y\rangle) = S$ and $\mathcal{L}(y) = \mathcal{B}(y) = \{C\}$

FIG. 4. Tableaux expansion rules for $\mathcal{SI}$

Theorem 4.7 *The tableaux algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.*

The dynamic blocking technique for $\mathcal{SI}$ and $\mathcal{SHI}$ described in Section 3, which is based on label equality, may lead to completion trees with exponentially long paths because there are exponentially many possibilities to label sets on such a path. Due to the non-deterministic $\sqcup$-rule, these exponentially many sets may actually occur.

This non-determinism is not problematical for $\mathcal{S}$ because disjunctions need not be completely decomposed to yield a subset-blocking situation. For an optimal $\mathcal{SI}$ algorithm, the additional label $\mathcal{B}$ was introduced to enable a sort of subset-blocking which is independent of the $\sqcup$-non-determinism. Intuitively, $\mathcal{B}(x)$ is the restriction of $\mathcal{L}(x)$ to those non-decomposed concepts that x must satisfy, whereas $\mathcal{L}(x)$ contains boolean decompositions of these concepts as well as those that are imposed by value restrictions in descendants. If x is blocked by y, then all concepts in $\mathcal{B}(x)$ are eventually decomposed in $\mathcal{L}(y)$ (if no clash occurs). However, in order to substitute x by y, x's constraints on predecessors must be at least as strong as y's; this is taken care of by the second blocking condition.

Let us consider a path $x_1, \ldots, x_n$ where all edges are labelled R with $\mathsf{Trans}(R)$, the only kind of paths along which the length of the longest concept in the labels might not decrease. If no rules can be applied, we have $\mathcal{L}(x_{i+1})/\mathsf{Inv}(R) \subseteq \mathcal{L}(x_i)/\mathsf{Inv}(R)$ and $\mathcal{B}(x_i) \subseteq \mathcal{B}(x_{i+1}) \cup \{C_i\}$ (where $\exists R.C_i$ triggered the generation of x_{i+1}). This limits the number of labels and guarantees blocking after a polynomial number of steps.

Lemma 4.8 *The paths of a completion tree for a concept D have a length of at most m^4 where $m = |sub(D)|$.*

Finally, a slight modification of the expansion rules given in Figure 4 yields a PSPACE algorithm. This modification is necessary because the original algorithm must keep the whole completion tree in its memory—which needs exponential space even though the length of its paths is polynomially bounded. The original algorithm may not forget about branches because restrictions which are pushed *upwards* in the tree might make it necessary to revisit paths which have been considered before. We solve this problem as follows:

Whenever the $\forall$- or the $\forall_+$-rule is applied to a node x and its *predecessor* y (Case 2' of these rules), we delete all successors of y from the completion tree. While this makes it necessary to restart the generation of successors for y, it makes it possible to implement the algorithm in a depth-first manner which facilitates the re-use of space.

This modification does not affect the proof of soundness and completeness for the algorithm, but we have to re-prove termination [27] as it relied on the fact that we never removed any nodes from the completion tree. Summing up we get:

Theorem 4.9 *The modified algorithm is a* PSPACE *decision procedure for satisfiability and subsumption of $\mathcal{SI}$-concepts.*

5 The Undecidability of Unrestricted $\mathcal{SHN}$

In [28] we describe an algorithm for $\mathcal{SHIQ}$ based on the $\mathcal{SHIF}$-algorithm already presented. Like earlier DLs that combine a hierarchy of (transitive and non-transitive) roles with some form of number restrictions [35; 27] and $\mathcal{SHIF}$, the DL $\mathcal{SHIQ}$ allows only *simple* roles in number restrictions. The justification for this limitation has been partly on the grounds of a doubtful semantics (of transitive functional roles) and partly to simplify decision procedures. In this section we will show that, even for the simpler $\mathcal{SHN}$ logic, allowing arbitrary roles in number restrictions leads to undecidability, while decidability for the corresponding variant of $\mathcal{SHIF}$ is still an open problem. For convenience, we will refer to $\mathcal{SHN}$ with arbitrary roles in number restrictions as $\mathcal{SHN}^+$.

The undecidability proof uses a reduction of the domino problem [7] adapted from [4]. This problem asks if, for a set of domino types, there exists a *tiling* of an $\mathbb{N}^2$ grid such that each point of the grid is covered with one of the domino types, and adjacent dominoes are "compatible" with respect to some predefined criteria.

Definition 5.1 *A domino system $\mathcal{D} = (D, H, V)$ consists of a non-empty set of domino types $D = \{D_1, \ldots, D_n\}$, and of sets of horizontally and vertically matching pairs $H \subseteq D \times D$ and $V \subseteq D \times D$. The problem is to determine if, for a given $\mathcal{D}$, there exists a* tiling *of an $\mathbb{N} \times \mathbb{N}$ grid such that each point of the grid is covered with a domino type in D and all horizontally and vertically adjacent pairs of domino types are in H and V respectively, i.e., a mapping $t : \mathbb{N} \times \mathbb{N} \to D$ such that for all $m, n \in \mathbb{N}$, $\langle t(m,n), t(m+1,n)\rangle \in H$ and $\langle t(m,n), t(m,n+1)\rangle \in V$.*

This problem can be reduced to the satisfiability of $\mathcal{SHN}^+$-concepts, and the undecidability of the domino problem implies undecidability of satisfiability of $\mathcal{SHN}^+$-concepts.

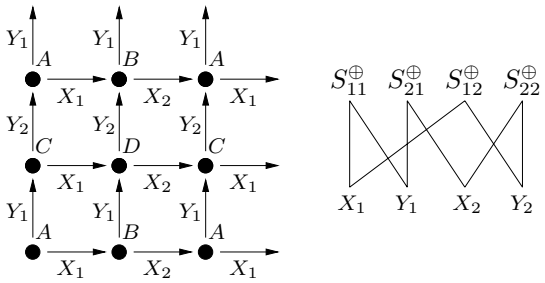


FIG. 5. Visualisation of the grid and role hierarchy.

Ensuring that a given point satisfies the compatibility conditions is simple for most logics (using value restrictions and boolean connectives), and applying such conditions throughout the grid is also simple in a logic such as $\mathcal{SHN}^+$ which can deal with arbitrary axioms. The crucial difficulty is representing the $\mathbb{N} \times \mathbb{N}$ grid using "horizontal" and "vertical" roles X and Y, and in particular forcing the coincidence of $X \circ Y$ and $Y \circ X$ successors. This can be accomplished in $\mathcal{SHN}^+$ using an alternating pattern of two horizontal roles X_1 and X_2, and two vertical roles Y_1 and Y_2, with disjoint primitive concepts A, B, C, and D being used to identify points in the grid with different combinations of successors. The coincidence of $X \circ Y$ and $Y \circ X$ successors can then be enforced using number restrictions on transitive super-roles of each of the four possible combinations of X and Y roles. A visualisation of the resulting grid and a suitable role hierarchy is shown in Figure 5, where $S^{\oplus}_{ij}$ are transitive roles.

The alternation of X and Y roles in the grid means that one of the transitive super-roles $S^{\oplus}_{ij}$ connects each point (x, y) to the points $(x+1, y)$, $(x, y+1)$ and $(x+1, y+1)$, and to no other points. A number restriction of the form $\leqslant 3 S^{\oplus}_{ij}$ can thus be used to enforce the necessary coincidence of $X \circ Y$ and $Y \circ X$ successors. A complete specification of the grid is given by the following axioms:

$$A \sqsubseteq \neg B \sqcap \neg C \sqcap \neg D \sqcap \exists X_1.B \sqcap \exists Y_1.C \sqcap \leqslant 3 S^{\oplus}_{11},$$
$$B \sqsubseteq \neg A \sqcap \neg C \sqcap \neg D \sqcap \exists X_2.A \sqcap \exists Y_1.D \sqcap \leqslant 3 S^{\oplus}_{21},$$
$$C \sqsubseteq \neg A \sqcap \neg B \sqcap \neg D \sqcap \exists X_1.D \sqcap \exists Y_2.A \sqcap \leqslant 3 S^{\oplus}_{12},$$
$$D \sqsubseteq \neg A \sqcap \neg B \sqcap \neg C \sqcap \exists X_2.C \sqcap \exists Y_2.B \sqcap \leqslant 3 S^{\oplus}_{22}.$$

It only remains to add axioms which encode the local compatibility conditions (as described in [4]) and to assert that A is subsumed by the disjunction of all domino types. The $\mathcal{SHN}^+$-concept A is now satisfiable w.r.t. the various axioms (which can be internalised as described in Lemma 2.3) iff there is a compatible tiling of the grid.

6 Implementation and Optimisation

The development of the $\mathcal{SI}$ family of DLs has been motivated by the desire to implement systems with good typical case performance. As discussed in Section 3, this is achieved in part through the design of the logics and algorithms themselves, in particular by using transitive roles and by reasoning with number restrictions directly, rather

than via encodings. Another important feature of these algorithms is that their relative simplicity facilitates the application of a range of optimisation techniques. Several systems based on $\mathcal{S}$ logics have now been implemented (e.g., FaCT [30], DLP [40] and RACE [23]), and have demonstrated that suitable optimisation techniques can lead to a dramatic improvement in the performance of the algorithms when used in realistic applications. A system based on the $\mathcal{SHIF}$ logic has also been implemented (iFaCT [32]) and has been shown to be similarly amenable to optimisation.

DL systems are typically used to classify a KB, and the optimisation techniques used in such systems can be divided into four categories based on the stage of the classification process at which they are applied.

1. Preprocessing optimisations that try to modify the KB so that classification and subsumption testing are easier.
2. Partial ordering optimisations that try to minimise the number of subsumption tests required in order to classify the KB.
3. Subsumption optimisations that try to avoid performing a potentially expensive satisfiability test, usually by substituting a cheaper test.
4. Satisfiability optimisations that try to improve the typical case performance of the underlying satisfiability testing algorithm.

Many optimisations in the first three categories are relatively independent of the underlying subsumption (satisfiability) testing algorithm and could be applied to any DL system. As we are mostly concerned with algorithms for the $\mathcal{SI}$ family of DLs we will concentrate on the fourth kind of optimisation, those that try to improve the performance of the algorithm itself. Most of these are aimed at reducing the size of the search space explored by the algorithm as a result of applying non-deterministic tableaux expansion rules.

6.1 Semantic Branching Search

Implementations of the algorithms described in the previous sections typically use a search technique called *syntactic branching*. When expanding the label of a node x, syntactic branching works by choosing an unexpanded disjunction $(C_1 \sqcup \ldots \sqcup C_n)$ in $\mathcal{L}(x)$ and searching the different models obtained by adding each of the disjuncts $C_1, \ldots, C_n$ to $\mathcal{L}(x)$ [22]. As the alternative branches of the search tree are not disjoint, there is nothing to prevent the recurrence of an unsatisfiable disjunct in different branches. The resulting wasted expansion could be costly if discovering the unsatisfiability requires the solution of a complex sub-problem. For example, tableaux expansion of a node x, where $\{(A \sqcup B), (A \sqcup C)\} \subseteq \mathcal{L}(x)$ and A is an unsatisfiable concept, could lead to the search pattern shown in Figure 6, in which the unsatisfiability of $\mathcal{L}(x) \cup \{A\}$ must be demonstrated twice.

This problem can be dealt with by using a *semantic branching* technique adapted from the Davis-Putnam-Logemann-Loveland procedure (DPL) commonly used to solve propositional satisfiability (SAT) problems [12; 21]. Instead of choosing an unexpanded disjunction in $\mathcal{L}(x)$, a single disjunct D is chosen from one of the unexpanded disjunctions in $\mathcal{L}(x)$. The two possible sub-trees obtained by adding either D or $\neg D$ to $\mathcal{L}(x)$ are then searched. Because the two sub-trees are strictly disjoint, there is no possibility of wasted search as in syntactic branching. Note that the order

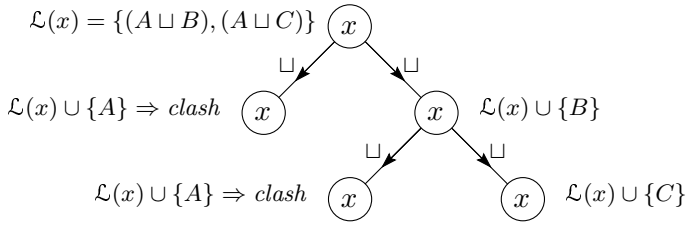


FIG. 6. Syntactic branching search

in which the two branches are explored is irrelevant from a theoretical viewpoint, but may offer further optimisation possibilities (see Section 6.4).

Semantic branching search has the additional advantage that a great deal is known about the implementation and optimisation of the DPL algorithm. In particular, both *local simplification* (see Section 6.2) and *heuristic guided search* (see Section 6.4) can be used to try to minimise the size of the search tree (although it should be noted that both these techniques can also be adapted for use with syntactic branching search).

There are also some disadvantages to semantic branching search. Firstly, it is possible that performance could be degraded by adding the negated disjunct in the second branch of the search tree, for example if the disjunct is a very large or complex concept. However this does not seem to be a serious problem in practice, with semantic branching rarely exhibiting significantly worse performance than syntactic branching. Secondly, its effectiveness is problem dependent. It is most effective with randomly generated problems, particularly those that are over-constrained (likely to be unsatisfiable) [34]. It is also effective with some of the hand crafted problems from the Tableaux'98 benchmark suite [24; 6]. However it is of little benefit when classifying realistic KBs [33].

6.2 Local Simplification

Local simplification is another technique used to reduce the size of the search space resulting from the application of non-deterministic expansion rules. Before any non-deterministic expansion of a node label $\mathcal{L}(x)$ is performed, disjunctions in $\mathcal{L}(x)$ are examined, and if possible simplified. The simplification most commonly used is to deterministically expand disjunctions in $\mathcal{L}(x)$ that present only one expansion possibility and to detect a clash when a disjunction in $\mathcal{L}(x)$ has no expansion possibilities. This simplification has been called *boolean constraint propagation* (BCP) [20]. In effect, the inference rule

$$\frac{\neg C_1, \ldots, \neg C_n, C_1 \sqcup \ldots \sqcup C_n \sqcup D}{D}$$

is being used to simplify the conjunctive concept represented by $\mathcal{L}(x)$. For example, given a node x such that

$$\{(C \sqcup (D_1 \sqcap D_2)), (\neg D_1 \sqcup \neg D_2 \sqcup C), \neg C\} \subseteq \mathcal{L}(x),$$

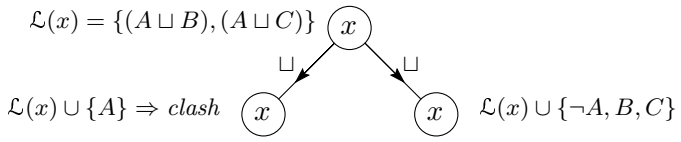


FIG. 7. Semantic branching search

BCP deterministically expands the disjunction $(C \sqcup (D_1 \sqcap D_2))$, adding $(D_1 \sqcap D_2)$ to $\mathcal{L}(x)$, because $\neg C \in \mathcal{L}(x)$. The deterministic expansion of $(D_1 \sqcap D_2)$ adds both D_1 and D_2 to $\mathcal{L}(x)$, allowing BCP to identify $(\neg D_1 \sqcup \neg D_2 \sqcup C)$ as a clash (without any branching having occurred), because $\{D_1, D_2, \neg C\} \subseteq \mathcal{L}(x)$.

BCP simplification is usually described as an integral part of SAT based algorithms [22], but it can also be used with syntactic branching. However, it is more effective with semantic branching as the negated concepts introduced by failed branches can result in additional simplifications. Taking the above example of $\{(A \sqcup B), (A \sqcup C)\} \subseteq \mathcal{L}(x)$, adding $\neg A$ to $\mathcal{L}(x)$ allows BCP to deterministically expand both of the disjunctions using the simplifications $(A \sqcup B)$ and $\neg A \rightarrow B$ and $(A \sqcup C)$ and $\neg A \rightarrow C$. The reduced search space resulting from the combination of semantic branching and BCP is shown in Figure 7.

Local simplification has the advantage that it can never increase the size of the search space and can thus only degrade performance to the extent of the overhead required to perform the simplification. Minimising this overhead does, however, require complex data structures [20], particularly in a modal/description logic setting.

As with semantic branching, effectiveness is problem dependent, the optimisation being most effective with over-constrained randomly generated problems [33].

6.3 Dependency Directed Backtracking

Inherent unsatisfiability concealed in sub-problems can lead to large amounts of unproductive backtracking search, sometimes called thrashing. For example, expanding a node x (using semantic branching), where

$$\mathcal{L}(x) = \{(C_1 \sqcup D_1), \ldots, (C_n \sqcup D_n), \exists R.(A \sqcap B), \forall R.\neg A\},$$

could lead to the fruitless exploration of 2^n possible R-successors of x before the inherent unsatisfiability is discovered. The search tree resulting from the tableaux expansion is illustrated in Figure 8.

This problem can be addressed by adapting a form of dependency directed backtracking called *backjumping*, which has been used in solving constraint satisfiability problems [5] (a similar technique was also used in the HARP theorem prover [39]). Backjumping works by labelling each concept in a node label with a dependency set indicating the branching points on which it depends. A concept $C \in \mathcal{L}(x)$ depends on a branching point if C was added to $\mathcal{L}(x)$ at the branching point or if $C \in \mathcal{L}(x)$ was generated by an expansion rule (including simplification) that depends on another concept $D \in \mathcal{L}(y)$, and $D \in \mathcal{L}(y)$ depends on the branching point. A concept $C \in \mathcal{L}(x)$ depends on a concept $D \in \mathcal{L}(y)$ when C was added to $\mathcal{L}(x)$ by a deterministic expansion that used $D \in \mathcal{L}(y)$. For example, if $A \in \mathcal{L}(x)$ was derived from the

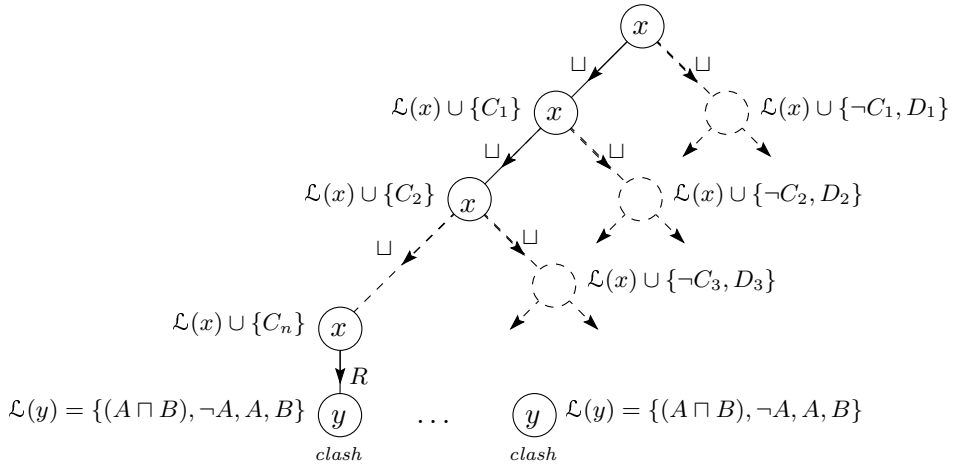


FIG. 8. Thrashing in backtracking search

expansion of $(A \sqcap B) \in \mathcal{L}(x)$, then $A \in \mathcal{L}(x)$ depends on $(A \sqcap B) \in \mathcal{L}(x)$.

When a clash is discovered, the dependency sets of the clashing concepts can be used to identify the most recent branching point where exploring the other branch might alleviate the cause of the clash. It is then possible to jump back over intervening branching points *without* exploring any alternative branches. Let us consider the earlier example and suppose that $\exists R.(A \sqcap B)$ has a dependency set $\mathbf{D}_i$ and $\forall R.\neg A$ has a dependency set $\mathbf{D}_j$. The search proceeds until $C_1 \ldots C_n$ have been added to $\mathcal{L}(x)$, when $\exists R.(A \sqcap B)$ and $\forall R.\neg A$ are deterministically expanded and a clash occurs in $\mathcal{L}(y)$ between the A derived from $\exists R.(A \sqcap B)$ and the $\neg A$ derived from $\forall R.\neg A$. As these derivations were both deterministic, the dependency sets will be $\mathbf{D}_i$ and $\mathbf{D}_j$ respectively, and so $\mathbf{D}_i \cup \mathbf{D}_j$ is returned. This set cannot include the branching points where $C_1 \ldots C_n$ were added to $\mathcal{L}(x)$ as $\mathbf{D}_i$ and $\mathbf{D}_j$ were defined before these branching points were reached. The algorithm can therefore backtrack through each of the preceding n branching points without exploring the second branches, and will continue to backtrack until it reaches the branching point equal to the maximum value in $\mathbf{D}_i \cup \mathbf{D}_j$ (if $\mathbf{D}_i = \mathbf{D}_j = \emptyset$, then the algorithm will backtrack through all branching points and return "unsatisfiable"). Figure 9 illustrates the pruned search tree, with the number of R-successors explored being reduced by an exponential number.

Backjumping can also be used with syntactic branching, but the procedure is slightly more complex as there may be more than two possible choices at a given branching point, and the dependency set of the disjunction being expanded must also be taken into account.

Like local simplification, backjumping can never increase the size of the search space. Moreover, it can lead to a dramatic reduction in the size of the search tree and thus a huge performance improvement. For example, when using either FaCT or DLP with backjumping disabled in order to classify a large (≈3,000 concept) KB derived from the European GALEN project [42], single satisfiability tests were encountered that could not be solved even after several weeks of CPU time. Classifying the same KB with backjumping enabled takes less than 100s of CPU time for either FaCT or

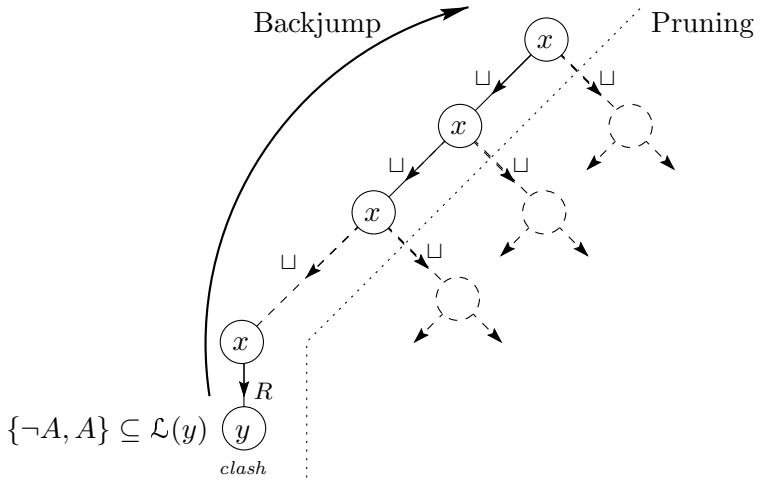


FIG. 9. Pruning the search using backjumping

DLP [34].

Backjumping's only disadvantage is the overhead of propagating and storing the dependency sets. This can be alleviated to some extent by using a pointer based implementation so that propagating a dependency set only requires the copying of a pointer.

6.4 Heuristic Guided Search

Heuristic techniques can be used to guide the search in a way that tries to minimise the size of the search tree. A method that is widely used in DPL SAT algorithms is to branch on the disjunct that has the *Maximum number of Occurrences in disjunctions of Minimum Size*—the well known MOMS heuristic [20]. By choosing a disjunct that occurs frequently in small disjunctions, the MOMS heuristic tries to maximise the effect of BCP. For example, if the label of a node x contains the unexpanded disjunctions $C \sqcup D_1, \ldots, C \sqcup D_n$, then branching on C leads to their deterministic expansion in a single step: when C is added to $\mathcal{L}(x)$, all of the disjunctions are fully expanded and when $\neg C$ is added to $\mathcal{L}(x)$, BCP will expand all of the disjunctions, causing $D_1, \ldots, D_n$ to be added to $\mathcal{L}(x)$. Branching first on any of $D_1, \ldots, D_n$, on the other hand, would only cause a single disjunction to be expanded.

The MOMS value for a candidate concept C is computed simply by counting the number of times C or its negation occur in minimally sized disjunctions. There are several variants of this heuristic, including the heuristic from Jeroslow and Wang [36]. The Jeroslow and Wang heuristic considers all occurrences of a disjunct, weighting them according to the size of the disjunction in which they occur. The heuristic then selects the disjunct with the highest overall weighting, again with the objective of maximising the effect of BCP and reducing the size of the search tree.

When a disjunct C has been selected from the disjunctions in $\mathcal{L}(x)$, a BCP maximising heuristic can also be used to determine the order in which the two possible

branches, $\mathcal{L}(x) \cup \{C\}$ and $\mathcal{L}(x) \cup \{\neg C\}$, are explored. This is done by separating the two components of the heuristic weighting contributed by occurrences of C and $\neg C$, trying $\mathcal{L}(x) \cup \{C\}$ first if C made the *smallest* contribution, and trying $\mathcal{L}(x) \cup \{\neg C\}$ first otherwise. The intention is to prune the search tree by maximising BCP in the first branch.

Unfortunately MOMS-style heuristics can interact adversely with the backjumping optimisation because they do not take dependency information into account. This was first discovered in the FaCT system, when it was noticed that using MOMS heuristic often led to much worse performance. The cause of this phenomenon turned out to be the fact that, without the heuristic, the data structures used in the implementation naturally led to "older" disjunctions (those dependent on earlier branching points) being expanded before "newer" ones, and this led to more effective pruning if a clash was discovered. Using the heuristic disturbed this ordering and reduced the effectiveness of backjumping [29].

Moreover, MOMS-style heuristics are of little value themselves in description logic systems because they rely for their effectiveness on finding the same disjuncts recurring in multiple unexpanded disjunctions: this is likely in hard propositional problems, where the disjuncts are propositional variables, and where the number of different variables is usually small compared to the number of disjunctive clauses (otherwise problems would, in general, be trivially satisfiable); it is unlikely in concept satisfiability problems, where the disjuncts are (possibly non-atomic) concepts, and where the number of different concepts is usually large compared to the number of disjunctive clauses. As a result, these heuristics will often discover that all disjuncts have similar or equal priorities, and the guidance they provide is not particularly useful.

An alternative strategy is to employ an *oldest-first* heuristic that tries to maximise the effectiveness of backjumping by using dependency sets to guide the expansion [34]. When choosing a disjunct on which to branch, the heuristic first selects those disjunctions that depend on the least recent branching points (i.e., those with minimal maximum values in their dependency sets), and then selects a disjunct from one of these disjunctions. This can be combined with the use of a BCP maximising heuristic, such as the Jeroslow and Wang heuristic, to select the disjunct from amongst the selected disjunctions.

The oldest-first heuristic can also be used to advantage when selecting the order in which existential role restrictions, and the labels of the R-successors which they generate, are expanded. One possible technique is to use the heuristic to select an unexpanded existential role restriction $\exists R.C$ from the label of a node x, apply the $\exists$-rule and the $\forall$-rule as necessary, and expand the label of the resulting R-successor. If the expansion results in a clash, then the algorithm will backtrack; if it does not, then continue selecting and expanding existential role restrictions from $\mathcal{L}(x)$ until it is fully expanded. A better technique is to first apply the $\exists$-rule and the $\forall$-rule exhaustively, creating a set of successor nodes. The order in which to expand these successors can then be based on the minimal maximum values in the dependency sets of all the concepts in their label, some of which may be due to universal role restrictions in $\mathcal{L}(x)$.

The main advantage of heuristics is that they can be used to complement other optimisations. The MOMS and Jeroslow and Wang heuristics, for example, are designed to increase the effectiveness of BCP while the oldest-first heuristic is designed

to increase the effectiveness of backjumping. They can also be selected and tuned to take advantage of the kinds of problem that are to be solved (if this is known). The BCP maximisation heuristics, for example, are generally quite effective with large randomly generated and hand crafted problems, whereas the oldest-first heuristic is more effective when classifying realistic KBs.

Unfortunately heuristics also have several disadvantages. They can add a significant overhead as the heuristic function may be expensive to evaluate and may need to be reevaluated at each branching point. Moreover, they may not improve performance, and may significantly degrade it, for example by interacting adversely with other optimisations, by increasing the frequency with which pathological worst cases can be expected to occur in generally easy problem sets.

6.5 Caching Satisfiability Status

During a satisfiability check there may be many successor nodes created. Some of these nodes can be very similar, particularly as the labels of the R-successors for a node x each contain the same concepts derived from the universal role restrictions in $\mathcal{L}(x)$. Systems such as DLP take advantage of this similarity by caching the satisfiability status of the sets of concepts with which node labels are initialised when they are created. The tableaux expansion of a node can then be avoided if the satisfiability status of its initial set of concepts is found in the cache.

However, this technique depends on the logic having the property that the satisfiability of a node is completely determined by its initial label set, and, due to the possible presence of inverse roles, $\mathcal{SI}$ logics do not have this property. For example, if the expansion of a node x generates an R-successor node y, with $\mathcal{L}(y) = \{\forall R^{-}.C\}$, then the satisfiability of y clearly also depends on the set of concepts in $\mathcal{L}(x)$. Similar problems could arise in the case where $\mathcal{L}(y)$ contains number restriction concepts.

If it is possible to solve these problems, then caching may be a very effective technique for $\mathcal{SI}$ logics, as it has been shown to be in the DLP system with a logic that does not support inverse roles. Caching is particularly useful in KB classification as cached values can be retained across multiple satisfiability tests. It can also be effective with both satisfiable and unsatisfiable problems, unlike many other optimisation techniques that are primarily aimed at speeding up the detection of unsatisfiability.

The main disadvantage with caching is the storage overhead incurred by retaining node labels (and perhaps additional information in the case of $\mathcal{SI}$ logics) and their satisfiability status throughout a satisfiability test (or longer, if the results are to be used in later satisfiability tests). An additional problem is that it interacts adversely with the backjumping optimisation as the dependency information required for backjumping cannot be effectively calculated for nodes that are found to be unsatisfiable as a result of a cache lookup. Although the set of concepts in the initial label of such a node is the same as that of the expanded node whose (un)satisfiability status has been cached, the dependency sets attached to the concepts that made up the two labels may not be the same. However, a weaker form of backjumping can still be performed by taking the dependency set of the unsatisfiable node to be the union of the dependency sets from the concepts in its label.

7 Discussion

A new DL system is being implemented based on the $\mathcal{SHIQ}$ algorithm we have developed from the $\mathcal{SHIF}$-algorithm described in Section 4.1 [28]. Pending the completion of this project, the existing FaCT system [31] has been modified to deal with inverse roles using the $\mathcal{SHIF}$ blocking strategy, the resulting system being referred to as iFaCT.

iFaCT has been used to conduct some initial experiments with a terminology representing (fragments of) database schemata and inter schema assertions from a data warehousing application [10] (a slightly simplified version of the proposed encoding was used to generate $\mathcal{SHIF}$ terminologies). iFaCT is able to classify this terminology, which contains 19 concepts and 42 axioms, in less than 0.1s of (266MHz Pentium) CPU time. In contrast, eliminating inverse roles using an embedding technique [11] gives an equisatisfiable FaCT terminology with an additional 84 axioms, but one which FaCT is unable to classify in 12 hours of CPU time. As discussed in Section 3, an extension of the embedding technique can be used to eliminate number restrictions [14], but requires a target logic which supports the transitive *closure* of roles, i.e., *converse*-PDL. The even larger number of axioms that this embedding would introduce makes it unlikely that tractable reasoning could be performed on the resulting terminology. Moreover, we are not aware of any algorithm for *converse*-PDL which does not employ a so-called *look behind analytical cut* [16], the application of which introduces considerable additional non-determinism. It seems inevitable that this would lead to a further degradation in empirical tractability.

The DL $\mathcal{SHIQ}$ will allow the above mentioned encoding of database schemata to be fully captured using qualified number restrictions. Future work will include completing the implementation of the $\mathcal{SHIQ}$ algorithm, testing its behaviour in this kind of application and investigating new techniques for improving its empirical tractability.

References

[1] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information*, 2:1–18, 1993.

[2] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[3] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-90-13, DFKI, Kaiserslautern, Deutschland, 1990. An abridged version appeared in *Proc. of IJCAI-91*, pp. 446–451.

[4] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proc. of KR'96*, pages 328–339. Morgan Kaufmann Publishers, San Francisco, California, November 1996.

[5] A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

[6] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics — introduction and summary. vol. 1397 of *LNAI*, pages 25–26. Springer-Verlag, 1998.

[7] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66, 1966.

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

[9] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of KR-94*, pages 109–120, Bonn, 1994. M. Kaufmann, Los Altos.

[10] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proc, of DEXA-98*, pages 192–197. IEEE Computer Society Press, 1998.

[11] D. Calvanese, G. De Giacomo, and R. Rosati. A note on encoding inverse roles and functional restrictions in $\mathcal{ALC}$ knowledge bases. In Franconi et al. [19].

[12] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[13] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. vol. 838 of *LNAI*, pages 332–346. Springer-Verlag, 1994.

[14] G. De Giacomo and M. Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of IJCAI-95*, pages 801–807, 1995.

[15] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proc. of KR-96*, pages 316–327. M. Kaufmann, Los Altos, 1996.

[16] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 1999. To appear.

[17] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Foundation of Knowledge Representation*. CSLI Publication, Cambridge University Press, 1996.

[18] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Science*, 18:194–211, 1979.

[19] E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors. *Proc. of DL'98*. CEUR, May 1998.

[20] J. W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[21] J. W. Freeman. Hard random 3-SAT problems and the Davis-Putnam procedure. *AIJ*, 81:183–198, 1996.

[22] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of CADE-96*, LNAI, New Brunswick, NJ, USA, 1996.

[23] V. Haarslev and R. Möller. RACE system description. In Lambrix et al. [38], pages 130–132.

[24] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, and S4. Technical report IAM-96-015, University of Bern, Switzerland, 1996.

[25] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of ECAI-90*, Pitman Publishing, London, 1990.

[26] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, Boston, MA, USA, 1991.

[27] I. Horrocks, U. Sattler, and S. Tobies. A PSPACE-algorithm for deciding $\mathcal{ALCI}_{R^+}$-satisfiability. Technical Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, 1998.

[28] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180, 1999.

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

[30] I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of Tableaux'98*, number 1397 in LNAI, pages 307–312. Springer-Verlag, Berlin, May 1998.

[31] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proc. of KR-98*, pages 636–647. Morgan Kaufmann Publishers, San Francisco, California, June 1998.

[32] I. Horrocks. FaCT and iFaCT. In Lambrix et al. [38], pages 133–135.

[33] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. In Franconi et al. [19], pages 90–94.

[34] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.

[35] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[36] R. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1:167–187, 1990.

[37] D. Kozen and J. Tiuryn. Logics of programs. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science – Formal Models and Semantics*, pages 789–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[38] P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors. *Proc. of DL'99*, 1999.

[39] F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *J. of Automated Reasoning*, 4:69–100, 1988.

[40] P. F. Patel-Schneider. DLP system description. In Franconi et al. [19], pages 87–89.

[41] V. R. Pratt. Models of program logic. *Proc. of FOCS-79*, pages 115–122, 1979.

[42] A. L. Rector, W. A. Nowlan, and A. Glowinski. Goals for concept representation in the GALEN project. In *Proc. of SCAMC'93*, pages 414–418, Washington DC, USA, 1993.

[43] U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für KI*, vol. 1137 of *LNAI*. Springer-Verlag, 1996.

[44] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, 1991.

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

[46] E. Spaan. The complexity of propositional tense logics. In M. de Rijke, editor, *Diamonds and Defaults*, pages 287–307. Kluwer Academic Publishers, Dordrecht, 1993.

Received February 15, 2000

Resolution-Based Methods for Modal Logics

HANS DE NIVELLE, *Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.*
E-mail: nivelle@mpi-sb.mpg.de

RENATE A. SCHMIDT, *Department of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom.*
E-mail: schmidt@cs.man.ac.uk

ULLRICH HUSTADT, *Centre for Agent Research and Development, Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom.*
E-mail: U.Hustadt@doc.mmu.ac.uk

Abstract

In this paper we give an overview of resolution methods for extended propositional modal logics. We adopt the standard translation approach and consider different resolution refinements which provide decision procedures for the resulting clause sets. Our procedures are based on ordered resolution and selection-based resolution. The logics that we cover are multi-modal logics defined over relations closed under intersection, union, converse and possibly complementation.

Keywords: modal logic, solvable classes, guarded fragment, description logic, inference methods, resolution, tableaux, model generation

1 Introduction

Modal logics are very popular and appear in various disguises in many areas of computer science, including knowledge representation, the field of logics of programs, computational linguistics and agent based systems. While decidability is an important criterion in many of these areas increasingly more expressive modal logics which allow complex relational parameters of modal operators are being used. Consider an example from knowledge representation and linguistics domains. Here the universes of frames contain arbitrary elements instead of worlds. If E denotes the eats relation and C is the set of cheeses, then $\langle E\rangle C$ can be interpreted as denoting the set of cheese eaters. An expression which requires complex relational parameters is the set of cheese lovers: $[\neg(E \wedge L)]\neg C$, where L denotes the likes relation. We have $x \in [\neg(E \wedge L)]\neg C$ iff for any $y \in C$, both $E(x, y)$ and $L(x, y)$ are true. In words, cheese lovers are people who eat and like every cheese. The meaning of $x \in [E \wedge L]C$ would be 'everything that x eats and likes is cheese'. These kinds of expressions can be formulated in the logics we consider in this paper.

We focus on subsystems of the multi-modal logic $K_{(m)}(\cap, \cup, ^{-}, \smile)$ which is defined

over families of relations closed under intersection, union, complementation and converse. $K_{(m)}(\cap, \cup, ^-, \smile)$ extends Boolean modal logic [17] with converse on relations. It encompasses very many standard modal logics such as K, KT, KD, KB, KTB, and KDB, their independent joins, as well as the basic tense logic K_t and logics of philosophical interest, such as logics expressing inaccessibility, sufficiency, or both necessity and sufficiency, see e.g. [18, 23, 24]. Certain forms of interactions, for example, inclusions among relations, are covered as well. $K_{(m)}(\cap, \cup, ^-, \smile)$ is related to the description logic $\mathcal{ALB}$ which was first described in [28] and contains a large class of well known description logics.

We concentrate on translation-based resolution methods for modal logics. This means that we take a modal formula, translate it into classical logic through the Kripke-semantics, and then apply some variant of resolution to it. Translation-based approaches are sometimes regarded as being inferior to tableaux-based approaches, or other special-purpose inference approaches. Arguably recent advances in the implementation of tableaux-based modal theorem provers make it harder to motivate the endeavour of translation into first-order logic. Another criticism often brought forward is the difficulty of reading resolution proofs (this is not true in general, see [28]). From our perspective the combination of translation and first-order resolution has a number of advantages, as this paper aims to show. Some obvious advantages of translation approaches are the following. Any modal logic which can be embedded into first-order logic can be treated. The translations are straightforward, and can be obtained in time $O(n \log n)$, so no engineering effort is needed here. For the resolution part, standard resolution provers can be used, or otherwise they can be used with small adaptations (for example, Bliksem [10], SPASS [40], and Otter [34]). The translation approach is generic, it can handle first-order modal logics, undecidable modal logics, for example, de Rijke's dynamic modal logic [11], and combinations of modal and non-modal logics. In all cases we can at least ensure soundness and completeness. For a large class of expressive modal and description logics, resolution provers provide decision procedures, and often the same refinements decide also first-order generalisations such as the guarded fragment or Maslov's class K [14, 26].

This paper gives an overview of different resolution refinements which provide decision procedures for first-order fragments corresponding to a variety of extended modal logics. We will focus on fragments induced by the standard relational translation of modal logics. Other translation methods exist but, as yet, it is not known how to treat modal logics with complex modal parameters within the context of these translation methods. Surveys of the different translation methods are Ohlbach [35, 36] and Ohlbach, Nonnengart and Gabbay [37].

Regardless as to which translation method is adopted, a crucial decision is the choice of a suitable refinement of the basic resolution calculus for first-order logic. Depending on our aims we have various options. Ordering refinements provide decision procedures for very expressive logics, while if we are interested in generating models for satisfiable formulae selection-based refinements (or hyperresolution) are more natural (Fermüller et al. [12, 13], Leitsch [30], Hustadt and Schmidt [28, 29]). We will describe three resolution decision procedures: an ordered resolution decision procedure for a class of clauses induced by $K_{(m)}(\cap, \cup, ^-, \smile)$ (Section 5), an ordering refinement combined with a selection function for the guarded fragment (Section 6), and a refinement which relies solely on the selection of negative literals for certain

extensions of $K_{(m)}(\cap, \cup, \smile)$ (Section 7). The latter refinement has the property that for many modal logics its derivations resemble those of tableaux calculi. As with tableaux-based procedures our selection-based procedure can be used for the automatic construction of finite models for satisfiable input formulae. In Section 8 we define a semantic tableaux calculus for the logic $K_{(m)}(\cap, \cup, \smile)$ which is derived from the selection-based resolution procedure. We also consider the relationship to single step prefixed tableaux calculi and prove a number of simulation results. Preliminary definitions are given in Sections 2, 3 and 4. Section 2 contains definitions of the notational conventions and basic concepts. Of particular importance is the structural transformation of formulae. Section 3 defines the syntax and semantics of the logic $K_{(m)}(\cap, \cup, ^{-}, \smile)$ and specifies the standard translation mapping into first-order logic. A general framework of ordered resolution and selection is described in Section 4.

This overview is based on the papers [14, 28, 29]. Some results have been improved and others are new. The definition of the class DL* in Section 5, generalises the class of DL-clauses from [28]. Section 7 includes a new complexity result. The results for extensions of $K_{(m)}(\cap, \cup, \smile)$ with frame properties are slightly more general than in [29]. The close correspondence between selection-based resolution (or hyperresolution) and special purpose tableaux calculi is also mentioned in [13, 28, 29]. A novelty are the tableaux calculi which we have been able to extract from the selection-based resolution procedure. These are related to calculi for the corresponding description logics [22, 21], but they do not compile relational formulae away.

2 Preliminary Definitions and Conventions

Throughout, our notational convention is the following: x, y, z are the letters reserved for first-order variables, s, t, u, v for terms, a, b for constants, f, g, h for function symbols, p, q, r for propositional symbols, and P, Q, R for predicate symbols. A is the letter reserved for atoms, L for literals, and C, D for clauses. For sets of clauses we use the letter N. The Greek letters φ, ψ, ϕ are reserved for modal or first-order formulae, and α, β, γ are reserved for relational formulae.

A *literal* is an atom or the negation of an atom. The former is said to be a *positive literal* and the latter a *negative literal*. If the predicate symbol of a literal has arity one (resp. two) then we call this literal a *unary literal* (resp. *binary literal*). A clause with one literal is a *unit clause* (or unit). If this literal is a unary (resp. binary) literal then the clause will be called a unary (resp. binary) unit clause. In this paper *clauses* are assumed to be sets of literals. The empty clause will be denoted by $\emptyset$. The components in the variable partition of a clause are called *split components*, that is, split components do not share variables. A clause which cannot be split further will be called a *maximally split clause*. A *positive* (resp. *negative*) clause contains only *positive* (resp. *negative*) literals.

Two formulae or clauses are said to be *variants* of each other if they are equal modulo variable renaming. Variant clauses are assumed to be equal.

The polarity of (occurrences of) modal or first-order subformulae is defined as usual: Any occurrence of a proper subformula of an equivalence has *zero polarity*. For occurrences of subformulae not below a '$\leftrightarrow$' symbol, an occurrence of a subformula has *positive polarity* if it is one inside the scope of an even number of (explicit or implicit) negations, and it has *negative polarity* if it is one inside the scope of an odd

number of negations.

For any first-order formula φ, if λ is the position of a subformula in φ, then $\varphi|_\lambda$ denotes the subformula of φ at position λ and $\varphi[\psi \mapsto \lambda]$ is the result of replacing $\varphi|_\lambda$ at position λ by ψ. The set of all the positions of subformulae of φ will be denoted by $\mathrm{Pos}(\varphi)$.

The *structural transformation*, also referred to as *renaming*, associates with each element λ of $\Lambda \subseteq \mathrm{Pos}(\varphi)$ a predicate symbol Q_λ and a literal $Q_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the free variables of $\varphi|_\lambda$, the symbol Q_λ does not occur in φ and two symbols Q_λ and $Q_{\lambda'}$ are equal only if $\varphi|_\lambda$ and $\varphi|_{\lambda'}$ are equivalent formulae.[1] Let

$$\mathrm{Def}^+_\lambda(\varphi) = \forall x_1 \ldots x_n \ (Q_\lambda(x_1, \ldots, x_n) \rightarrow \varphi|_\lambda) \quad \text{and}$$
$$\mathrm{Def}^-_\lambda(\varphi) = \forall x_1 \ldots x_n \ (\varphi|_\lambda \rightarrow Q_\lambda(x_1, \ldots, x_n)).$$

The *definition* of Q_λ is the formula

$$\mathrm{Def}_\lambda(\varphi) = \begin{cases} \mathrm{Def}^+_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has positive polarity,} \\ \mathrm{Def}^-_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has negative polarity,} \\ \mathrm{Def}^+_\lambda(\varphi) \wedge \mathrm{Def}^-_\lambda(\varphi) & \text{otherwise.} \end{cases}$$

The corresponding clauses will be called *definitional clauses*. Now, define $\mathrm{Def}_\Lambda(\varphi)$ inductively by:

$$\mathrm{Def}_\emptyset(\varphi) = \varphi \quad \text{and}$$
$$\mathrm{Def}_{\Lambda \cup \{\lambda\}}(\varphi) = \mathrm{Def}_\Lambda(\varphi[Q_\lambda(x_1, \ldots, x_n) \mapsto \lambda]) \wedge \mathrm{Def}_\lambda(\varphi),$$

where λ is maximal in $\Lambda \cup \{\lambda\}$ with respect to the prefix ordering on positions. A *definitional form* of φ is $\mathrm{Def}_\Lambda(\varphi)$, where Λ is a subset of all positions of subformulae (usually, non-atomic or non-literal subformulae).

Theorem 2.1 (e.g. Plaisted and Greenbaum [39]) *Let φ be a first-order formula.*

1. *φ is satisfiable iff $\mathrm{Def}_\Lambda(\varphi)$ is satisfiable, for any $\Lambda \subseteq \mathrm{Pos}(\varphi)$.*
2. *$\mathrm{Def}_\Lambda(\varphi)$ can be computed in polynomial time.*

3 The Modal Logic $\boldsymbol{K}_{(m)}(\cap, \cup, ^-, \smile)$

$K_{(m)}(\cap, \cup, ^-, \smile)$ is the multi-modal logic defined over families of binary relations closed under intersection, union, complementation and converse.

The language of $K_{(m)}(\cap, \cup, ^-, \smile)$ is defined over countably many propositional variables $p, p_1, p_2, \ldots$, and countably many relational variables $r, r_1, r_2, \ldots$. A *propositional atom* is a propositional variable, $\top$ or $\bot$. A *modal formula* is either a propositional atom or a formula of the form $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$, where φ is a modal formula and α is a relational formula. A *relational formula* is a relational variable or has one of the following forms: $\alpha \wedge \beta$, $\alpha \vee \beta$, $\neg\alpha$, and $\alpha^\smile$ (converse), where α and β are relational formulae. Other connectives are defined to be abbreviations,

[1] In practice, one may want to use the same symbols for variant subformulae, or subformulae which are obviously equivalent, for example, $\varphi \vee \varphi$ and φ.

for example, $\varphi \rightarrow \psi = \neg\varphi \vee \psi$ or the universal modality is $[*] = [r_j \vee \neg r_j]$, for some relational variable r_j.

We will also consider logics with fewer relational operations. Formally, by a logic in-between K and $K_{(m)}(\cap, \cup, ^-, \smile)$ we mean a logic $K_{(m)}(\star_1, \ldots, \star_k)$ where $m \geq 1$, $1 \leq k \leq 4$ and the $\star_i$ are distinct operations from $\{\cap, \cup, ^-, \smile\}$.

The semantics of $K_{(m)}(\cap, \cup, ^-, \smile)$ is defined in terms of relational structures or frames. A frame is a tuple (W, R) of a non-empty set W (of worlds) and a mapping R from relational formulae to binary relations over W satisfying:

$$R(\alpha \wedge \beta) = R(\alpha) \cap R(\beta) \qquad R(\neg\alpha) = \overline{R(\alpha)}$$
$$R(\alpha \vee \beta) = R(\alpha) \cup R(\beta) \qquad R(\alpha^\smile) = R(\alpha)^\smile.$$

The defining class of frames of a modal logic determines, and is determined by, a corresponding class of models. A model (an interpretation) is given by a triple $\mathcal{M} = (W, R, \iota)$, where (W, R) is a frame and ι is a mapping from modal formulae to subsets of W satisfying:

$$\iota(\bot) = \emptyset \qquad \iota(\top) = W \qquad \iota(\neg\varphi) = \overline{\iota(\varphi)}$$
$$\iota(\varphi \wedge \psi) = \iota(\varphi) \cap \iota(\psi) \qquad \iota(\langle\alpha\rangle\varphi) = \{x \mid \exists y \in W\ (x, y) \in R(\alpha) \wedge y \in \iota(\varphi)\}$$
$$\iota(\varphi \vee \psi) = \iota(\varphi) \cup \iota(\psi) \qquad \iota([\alpha]\varphi) = \{x \mid \forall y \in W\ (x, y) \in R(\alpha) \rightarrow y \in \iota(\varphi)\}.$$

A modal formula φ is satisfiable if an $\mathcal{M}$ exists such that for some x in W, $x \in \iota(\varphi)$.

The standard translation of $K_{(m)}(\cap, \cup, ^-, \smile)$ into first-order logic follows the semantic definition and is therefore given by the following.

$$\pi(\top, x) = \top \qquad \pi(\bot, x) = \bot$$
$$\pi(p_i, x) = P_i(x) \qquad \pi(\neg\varphi, x) = \neg\pi(\varphi, x)$$
$$\pi(\varphi \star \psi, x) = \pi(\varphi, x) \star \pi(\psi, x) \quad \text{for } \star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$
$$\pi(\langle\alpha\rangle\varphi, x) = \exists y\,(\tau(\alpha, x, y) \wedge \pi(\varphi, y)) \qquad \pi([\alpha]\varphi, x) = \forall y\,(\tau(\alpha, x, y) \rightarrow \pi(\varphi, y)).$$

Relational formulae are translated according to:

$$\tau(r_j, x, y) = R_j(x, y)$$
$$\tau(\neg\alpha, x, y) = \neg\tau(\alpha, x, y) \qquad \tau(\alpha^\smile, x, y) = \tau(\alpha, y, x)$$
$$\tau(\alpha \star \beta, x, y) = \tau(\alpha, x, y) \star \tau(\beta, x, y) \quad \text{for } \star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

In the translation each propositional or relational variable (p_i or r_j) is uniquely associated with a unary or binary predicate variable, denoted by the corresponding capital letter (P_i or R_j).

By definition, Π maps any modal formula φ to $\exists x\ \pi(\varphi, x)$.

Theorem 3.1 *Let L be a logic in-between K and $K_{(m)}(\cap, \cup, ^-, \smile)$. For any modal formula φ, φ is satisfiable in L iff $\Pi(\varphi)$ is first-order satisfiable.*

In order to keep the presentation simple, modal formulae are assumed to be in negation normal form. This means that in every subformula of the form $\neg\varphi$, φ is a propositional variable. The negation normal form of any modal formula is obtained as usual, namely, by moving negation symbols inwards as far as possible (using De Morgan's laws, $\neg\langle\alpha\rangle\psi \leftrightarrow [\alpha]\neg\psi$ and $\neg[\alpha]\psi \leftrightarrow \langle\alpha\rangle\neg\psi$, and $\neg(\alpha^\smile) \leftrightarrow (\neg\alpha)^\smile$) and eliminating double negations.

4 The Resolution Framework

In this paper we will make use of A-ordered resolution, extended with selection. A-ordered resolution is well-known and widely used in resolution decision procedures [12, 13, 5, 34, 30, 26]. It follows from the results in Bachmair and Ganzinger [3, 4] that A-ordered resolution can be combined with a selection function. This selection function can override the A-ordering, give preference to inferences with negative literals. A-ordered resolution with selection is controlled by two parameters: an A-ordering and a selection function. An A-ordering is an ordering $\succ$ on atoms, which satisfies the following condition: For all atoms A, B and for all substitutions σ, $A \succ B$ implies $A\sigma \succ B\sigma$. For a literal $L = (\neg)A$ let $\mathrm{at}(L) = A$. A-orderings are extended to literals by $L \succ L'$ iff $\mathrm{at}(L) \succ \mathrm{at}(L')$. If one uses orderings that do not ignore the negation sign (these are called L-orderings), one does not loose completeness [7]. However L-orderings cannot be combined with selection. Given an A-ordering $\succ$, we define the *maximal literals* in a clause in the standard way: A literal L in a clause C is maximal in C, if there is no literal L' in C, for which $L' \succ L$.

Let $\succ$ be an A-ordering. A *selection function* S, based on $\succ$, is a function which assigns to each clause C a non-empty set of its literals, such that one of the following holds:

(4.1) Either $S(C)$ contains a negative literal, or

(4.2) $S(C)$ contains all the $\succ$-maximal literals of C.

No further restrictions are imposed on the selection function. If the selection function always prefers the second alternative, one has just A-ordered resolution. If the selection function always selects only the negative literals in non-positive clauses, then the restriction simulates A-ordered hyperresolution. Based on a selection function S, resolution and factoring can be defined as follows:

Resolution:
$$\frac{C \vee A_1 \quad \neg A_2 \vee D}{(C \vee D)\sigma}$$

provided (i) σ is the most general unifier of A_1 and A_2, and (ii) $A_1 \in S(C \vee A_1)$ and $\neg A_2 \in S(\neg A_2 \vee D)$. Then the clause $(C \vee D)\sigma$ is a *resolvent*.

Factoring:
$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

provided (i) σ is the most general unifier of A_1 and A_2, and (ii) $A_1 \in S(C \vee A_1 \vee A_2)$. Then the clause $(C \vee A_1)\sigma$ is called a *factor* of $C \vee A_1 \vee A_2$.

The combination of selection-based resolution and factoring forms a complete refutation system for clause sets.

The premise $C \vee A_1$ of the resolution rule and premise of the factoring rule will be referred to as a *positive premise*, while the premise $\neg A_2 \vee D$ of the resolution rule will be referred to as a *negative premise*. The literals resolved upon and factored upon are called *eligible literals*.

Simplification and Splitting

In the previous section we explained where the clauses come from. In this section we explain how to get rid of them. In order to obtain termination, one needs redundancy criteria. Let C and D be clauses. Clause C *subsumes* D if $|C| \leq |D|$, and there exists a substitution σ, such that $C\sigma \subseteq D$. Without the length-restriction factors would be subsumed by their parents. This would result in deletion of all factors. Since the factoring rule is necessary to completeness, deleting all factors would result in incompleteness. Determining whether or not clause C subsumes clause D, is NP-complete. A *condensation* of C is a minimal subset D of C, such that D subsumes C. One can show that condensations are unique up to renaming. Determining whether or not a clause is condensed, is NP-complete. Computing the condensation is NP-hard. In practice, NP-hardness does not cause problems, since the clauses are short ($< \log\log$) in comparison to the number of clauses. A clause C is a *tautology* if it contains a complementary pair of literals A and $\neg A$.

Let N be a clause set. A *saturation* of N is a clause set N_∞, such that, for every non-tautological clause C in N, there is a clause D in N_∞, such that D subsumes C, and for each non-tautological clause C, that is derivable from clauses in N_∞, there is a clause D in N_∞, such that D subsumes C.

For selection based resolution the following holds.

Theorem 4.1 *For every clause set N, and every saturation N_∞ of N the following holds: N is unsatisfiable iff N_∞ contains the empty clause.*

This follows from the results in Bachmair and Ganzinger [3, 4]. This completeness allows us to freely delete tautologies and subsumed clauses, or replace clauses by condensations. In general it is possible to use stronger notions of redundancy. One can define a clause to be redundant if it is implied by a finite set of strictly smaller clauses (under an appropriate extension of $\succ$ to clauses), see [3, 4].

Our notion of saturation is not appropriate for building into a real theorem prover, because it does not model the time aspect. A clause may become redundant only after some time, after it has been used for deriving clauses that occur in the proof.

The *splitting rule* is a rule that is borrowed from semantic tableaux. Let N be a set of clauses containing a clause C, that has two split components C_1 and C_2. Then, instead of trying to refute N one tries to refute $N \cup \{C_1\}$ and $N \cup \{C_2\}$ (or $N \cup \{C_1\}$ and $N \cup \{C_2, \neg C_1\}$, if C_1 is a ground clause). Note that in both sets, the original clause C has become redundant. The splitting rule can be essentially simulated in the resolution context by introducing a new propositional symbol. If $C_1 \vee C_2$ is a clause that can be split into two split components C_1 and C_2, then it is possible to replace $C_1 \vee C_2$ by two clauses $C_1 \vee q$, and $\neg q \vee C_2$. q is made minimal in the A-ordering, and $\neg q$ is selected. In most cases this is easier to implement than the full splitting rule.

5 Ordered Resolution for $\boldsymbol{K}_{(m)}(\cap, \cup, ^-, \smile)$

Many modal logics naturally translate into decidable fragments of first-order logic. For example the basic logic K translates into the two-variable fragment, and into the guarded fragment. By constructing decision procedures for these decidable fragments, one obtains generic decision procedures for modal logics. We consider two classes. One

is a clause fragment based on the two-variable fragment, called DL*. This fragment is a variation of the class of DL-clauses, that was introduced in Hustadt and Schmidt [28] with the purpose of handling expressive description logics. The other one is the guarded fragment, which was introduced by Andréka, Van Benthem and Németi [2] as the 'modal subset of first-order logic'. Although it did not quite meet the ambitious goals, it is an important fragment, containing many modal logics.

The class of DL*-clauses is related to the class $\mathcal{S}^+$ in Fermüller et al. [12]. This class was introduced there as the clause fragment belonging to the two-variable fragment. The class $\mathcal{S}^+$ can only be decided by a non-liftable ordering [8], or by an A-ordering combined with a rule called monadisation [12]. Since we try to root our approach on the common basis of liftable orderings, we slightly restrict the class, so that it can be decided by a liftable ordering. The restriction is still general enough to contain the clause translations of the Π-transformation of the modal formulae in $K_{(m)}(\cap, \cup, ^-, \smile)$.

We now introduce the clause fragment DL*. In order to simplify the exposition, we assume that all clauses are maximally split. The notions can be easily adopted for clauses with more than one split component.

Let C be a clause. It is a *DL*-clause* if

1. all literals are unary, or binary,
2. there is no nesting of function symbols,
3. every functional term in C contains all the variables of C, and
4. every binary literal (even if it has no functional terms) contains all variables of C.

Observe that 3. implies that if C contains a functional ground term, then C is ground. The difference with $\mathcal{S}^+$ is Condition 4. For $\mathcal{S}^+$, Condition 4 would be (4a): Every clause C has a literal containing all variables of C. Condition 4 forbids the following problematic clauses, which are allowed by Condition 4a: $P(x,x) \vee Q(x,y)$ and $\neg P(x,x) \vee R(x,y)$. In order to stay within $\mathcal{S}^+$, one would have to block the inference based on $P(x,x)$ and $\neg P(x,x)$, since this would result in the clause $Q(x,y) \vee R(x,z)$, which contains more variables than each of the parent clauses. However no A-ordering can put $Q(x,y) \succ P(x,x)$, for all predicate symbols P and Q.

Examples of DL*-clauses include ground clauses, and

$$\neg Q_0(x) \vee Q_1(x) \vee \neg Q_2(x) \qquad Q_0(x) \vee \neg R_0(x,y) \vee Q_1(y)$$
$$\neg Q_0(x) \vee Q_1(f(x)) \qquad \neg Q_0(x) \vee \neg R_0(f(x),x)$$
$$R_0(x,y) \vee \neg R_1(y,x) \vee R_2(x,y).$$

The clauses $R_0(x,y) \vee R_0(x,f(x))$, $Q_0(x,x,x) \vee Q_1(f(f(x)))$ and $R_0(x,x) \vee R_1(x,y)$ do not belong to the class of DL*-clauses. The clause $Q_0(x) \vee Q_1(a)$ does in principle belong to DL*, but is not maximally split.

Theorem 5.1 *Over a finite signature*[2] *there are only finitely many maximally split DL*-clauses (modulo variable renaming).*

The proof is similar to the proof for the class of DL-clauses in Hustadt and Schmidt [28]. The proof can be obtained by first observing that there is a fixed upper bound for

[2] The supply of function symbols and predicate symbols is finite, while there are possibly infinite but countably many variables.

the maximal number of variables in a clause. Then there are only a finite number of possible literals. Because every clause is a subset of the set of possible literals, there is a finite set of possible clauses.

Theorem 5.2 *The number of possible DL*-clauses is bounded by* $2^{2^{f(s)}}$, *where* f *is of order* $s\log(s)$ *and* s *is the size of the signature.*

PROOF. Let a be maximal arity of any function symbol. Because any clause contains at most a variables, the number of possible terms is bounded by $(s+a)+(s+a)^{a+1} \leq (s+a)^{a+2}$. The number of possible atoms is then equal to $s((s+a)^{a+2})^2 \leq (s+a)^{2a+5}$. The number of possible literals equals $2(s+a)^{2a+5} \leq (s+a)^{2a+6}$. Consequently, the number of non-equivalent clauses is bounded by $2^{(s+a)^{2a+6}} = 2^{2^{(2a+6)\log(s+a)}}$. ∎

The reduction of modal formulae to sets of DL*-clauses makes use of a structural transformation introducing new names for subformulae corresponding to non-literal subformulae of the original modal formula. For a given modal formula φ and its translation into first-order logic $\varphi' = \Pi(\varphi)$, we apply the mapping Def_Λ with

$$\Lambda = \{\lambda \mid \text{there is a non-literal subformula } \varphi|_{\lambda'} \text{ of } \varphi \text{ and } \varphi'|_\lambda = \Pi(\varphi|_{\lambda'})\}.$$

For example, the definition corresponding to a subformula $\langle r_j\rangle p$ is

$$\forall x\,(Q_{\langle r_j\rangle p}(x) \rightarrow \exists y\ (R_j(x,y) \wedge P(y))).$$

The formula

$$\exists x \forall y\,((\neg R_1(x,y) \wedge R_2(x,y)) \rightarrow \exists z\,(\neg R_1(y,z) \wedge R_2(y,z) \wedge P(z))), \qquad (*)$$

which is a translation of the modal formula $[\neg r_1 \wedge r_2]\langle \neg r_1 \wedge r_2\rangle p$ results in the following set of definitions, together with $\exists x\, Q_{[\alpha]\langle\alpha\rangle p}(x)$.

$$\begin{aligned}
&\forall x\,(Q_{[\alpha]\langle\alpha\rangle p}(x) \rightarrow \forall y\,(Q_\alpha(x,y) \rightarrow Q_{\langle\alpha\rangle p}(y)))\\
&\forall x\,(Q_{\langle\alpha\rangle p}(x) \rightarrow \exists y\,(Q_\alpha(x,y) \wedge P(y)))\\
&\forall xy\,(Q_\alpha(x,y) \rightarrow (\neg R_1(x,y) \wedge R_2(x,y)))\\
&\forall xy\,((\neg R_1(x,y) \wedge R_2(x,y)) \rightarrow Q_\alpha(x,y)).
\end{aligned}$$

Here α is used as an abbreviation for $\neg r_1 \wedge r_2$. Notice that one new symbol Q_α was used for the positive and negative occurrences of the subformula $\neg R_1(x,y) \wedge R_2(x,y)$.

Theorem 5.3 *Let* φ' *be a first-order formula that results from the translation of a modal formula* φ *in* $K_{(m)}(\cap, \cup, ^-, \smile)$. *Every clause in the clausal normal form of* $\mathrm{Def}_\Lambda(\varphi')$ *is a DL*-clause.*

PROOF. Not difficult. ∎

In order to decide the class DL*, we use the following A-ordering which is similar to the recursive path ordering. First we define an order $>_d$ on terms: $s >_d t$ if s is deeper than t, and every variable that occurs in t, occurs deeper in s. Then we define $P(s_1,\ldots,s_n) \succ Q(t_1,\ldots,t_m)$ as $\{s_1,\ldots,s_n\} >_d^{\mathrm{mul}} \{t_1,\ldots,t_m\}$. Here $>_d^{\mathrm{mul}}$ is the multiset extension of $>_d$. So we have $P(f(x)) \succ P(a), P(x)$ and $P(x,y) \succ Q(x)$, but

not $P(f(x)) \succ P(f(a))$. The $>_d$ ordering originates from Fermüller et al. [12]. The selection function S is completely determined by $\succ$, so there is no preferred selection of negative literals.

We now give the clausal normal form of the formula (*) above. The maximal literals are marked with *. These are the literals that can potentially be resolved or factored upon.

$$
\begin{aligned}
& Q_{[\alpha]\langle\alpha\rangle p}(a)^* \\
& \neg Q_{[\alpha]\langle\alpha\rangle p}(x) \vee \neg Q_\alpha(x,y)^* \vee Q_{\langle\alpha\rangle p}(y) \\
& \neg Q_{\langle\alpha\rangle p}(x) \vee Q_\alpha(x, f(x))^* \\
& \neg Q_{\langle\alpha\rangle p}(x) \vee P(f(x))^* \\
& \neg Q_\alpha(x,y)^* \vee \neg R_1(x,y)^* \\
& \neg Q_\alpha(x,y)^* \vee R_2(x,y)^* \\
& R_1(x,y)^* \vee \neg R_2(x,y)^* \vee Q_\alpha(x,y)^*
\end{aligned}
$$

In the last three clauses there is more than one maximal literal. This could be prevented by completing $\succ$ with an ordering on atoms. In that case it is necessary to distinguish equivalent from incomparable literals. Instead of $\succ$, one would have to define $\succeq$. Then $A \succ B$ would have to be defined as $A \succeq B$ and $A \not\preceq B$. In the case that $A \succeq B$ and $A \preceq B$, one can try to use a second ordering for establishing a priority.

In order to prove that the procedure that we described is indeed a decision procedure we have to show that it is complete, and terminating. The completeness follows from Theorem 4.1. Termination is a consequence of Theorem 5.1, and the fact that the restriction derives only clauses that are within DL^*, or that can be split. This fact is obtained by a case analysis, similar as in [28]. Therefore:

Theorem 5.4 *Let L be a logic in-between K and $K_{(m)}(\cap, \cup, ^-, \smile)$. Let N be the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$, where φ is any modal formula in L. Then:*

1. *Any derivation from N terminates in double exponential time.*
2. *φ is unsatisfiable in L iff the saturation of N contains the empty clause.*

This result covers actually a larger class of modal logics. Boolean modal logic, and hence also $K_{(m)}(\cap, \cup, ^-, \smile)$, is expressive enough to allow for frame properties to be specified by relational formulae. Implication of relational formulae can be defined by $(\alpha \to \beta) = [\alpha \wedge \neg\beta]\bot$ [38]. Hence, the symmetry of the accessibility relation R_1 associated with r_1 can be specified by $r_1 \to r_1^\smile$.

If Δ is a set of relational frame properties then $L\Delta$ will denote the logic characterised by the class of frames satisfying the conjunction of properties in Δ.

Corollary 5.5 *Let L be a logic in-between K and $K_{(m)}(\cap, \cup, ^-, \smile)$. Let Δ be the Boolean combination of relational inclusions or equivalences expressed over intersection, union, complementation and converse. Suppose φ is any modal formula and N is the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$. Then:*

1. *Any derivation from $N \cup \Delta$ terminates in double exponential time.*
2. *φ is unsatisfiable in $L\Delta$ iff the saturation of $N \cup \Delta$ contains the empty clause.*

The decidability result for the classes DL* and DL allows for a slightly more general result, which includes reflexivity and irreflexivity. Modal and relational formulae with positive occurrences of relational composition can also be embedded into the class DL*. Moreover, relational properties such as $\forall xy\,(R_1(x,y) \rightarrow R_2(x,x))$ are covered by the class $\mathcal{S}^+$.

6 Ordered Resolution for the Guarded Fragment

In this section we use ordered resolution with selection as a decision procedure for the guarded fragment. The guarded fragment was first shown decidable by Andréka, Németi and Van Benthem [1]. Grädel [20] has shown that the satisfiability problem for the guarded fragment is DEXPTIME-complete. There it was also shown that the guard condition is necessary only for the universal quantifiers, when the formula is in negation normal form. A resolution decision procedure for the guarded fragment was first established in de Nivelle [9]. In Ganzinger and de Nivelle [14] the method was adapted to the guarded fragment with equality. It is shown there that the complexity of the resolution decision procedure is consistent with the complexity given in [20]. The decision procedure that we give here is based on the one in [14].

A first-order formula is in the *guarded fragment* if it is function free, and every quantification has form $\forall\overline{x}\,(G \rightarrow \psi)$, or $\exists\overline{x}\,(G \wedge \psi)$. Here G is an atom containing all free variables of ψ, and $\overline{x}$ is a sequence of variables.

We use the following clausal normal form. A clause C is a *guarded clause* if

1. there is no nesting of function symbols,
2. every functional term in C contains all variables of C, and
3. if C contains variables, then there is a negative, function-free literal that contains all variables of C. Such a literal is called a *guard* literal.

As is the case with the class of DL*-clauses, there is only a finitely bounded set of guarded clauses.

Theorem 6.1 (Ganzinger and de Nivelle [14]) *Over a finite signature the number of possible guarded clauses is of order 2^{2^s}, where s is the size of the signature.*

For the reduction to clausal normal form we assume that a guarded formula φ is in negation normal form. The reduction of φ into guarded clauses uses a structural transformation Def_Λ with

$$\Lambda = \{\lambda \mid \lambda \text{ is a position in } \varphi \text{ of a formula of the form } \forall\overline{x}\,(G \rightarrow \psi)\}.$$

It can be shown that this structural transformation preserves the guarded fragment. The definitional formula that defines a guarded formula $\forall\overline{x}\,(G \rightarrow \psi)$, has the form

$$\forall\overline{y}\,(Q_{\forall\overline{x}(G\rightarrow\psi)}(\overline{y}) \rightarrow \forall\overline{x}\,(G \rightarrow \psi)).$$

Every variable in $\overline{y}$ and $\overline{x}$ occurs in G. This formula is not guarded by itself but it is equivalent to the following formula, which is guarded: $\forall\overline{xy}\,(G \rightarrow (Q_{\forall\overline{x}(G\rightarrow\psi)}(\overline{y}) \rightarrow \psi))$.

Formulae in $K_{(m)}(\cap,\cup,\smile)$ are translated by Π into the guarded fragment. Negations of accessibility relations would be problematic. For example, $[\neg r]p$ is translated into $\exists x\forall y\,(\neg R(x,y) \rightarrow P(y))$. This formula is not guarded. The formula

$[(r_1 \wedge r_2) \vee r_3^{\smile}]p$ is translated into $\exists x \forall y\,((R_1(x,y) \wedge R_2(x,y)\ ) \vee R_3(y,x) \rightarrow P(y))$. This formula is not guarded either, however, it is equivalent to the guarded formula:

$$\exists x \forall y\,(R_1(x,y) \rightarrow (R_2(x,y) \rightarrow P(y))) \wedge \forall y\,(R_3(y,x) \rightarrow P(y)).$$

We show that this is in general the case for formulae in $K_{(m)}(\cap, \cup, \smile)$. The mapping Π translates formulae of $K_{(m)}(\cap, \cup, \smile)$ into first-order formulae in which the quantifications have the form $\forall \overline{x}\,(G \rightarrow \psi)$. In this, G is a relational expression without negation and function symbols, in which each atom contains all free variables of ψ. This G can be translated into disjunctive normal form,

$$(G_{1,1} \wedge \ldots \wedge G_{1,l_1}) \vee \ldots \vee (G_{n,1} \wedge \ldots \wedge G_{n,l_n}).$$

The $G_{i,j}$ are atoms, containing all free variables of ψ. Then $\forall \overline{x}\,(G \rightarrow \psi)$ is equivalent to

$$\forall \overline{x}\,((G_{1,1} \wedge \ldots \wedge G_{1,l_1}) \rightarrow \psi) \wedge \ldots \wedge \forall \overline{x}\,((G_{n,1} \wedge \ldots \wedge G_{n,l_n}) \rightarrow \psi),$$

which is in turn equivalent to

$$\forall \overline{x}\,(G_{1,1} \rightarrow (\ldots \rightarrow (G_{1,l_1} \rightarrow \psi))) \wedge \ldots \wedge \forall \overline{x}\,(G_{n,1} \rightarrow (\ldots \rightarrow (G_{n,l_n} \rightarrow \psi))).$$

The $G_{i,1}$ are well-formed guards.

In order to obtain a decision procedure for the guarded fragment, we make use of the ordering $\succ$ of the previous section, combined with selection of negative literals.

1. If C is a non-ground clause without functional terms, then $S(C)$ contains all guards of C.
2. If C is a clause with functional terms, then $S(C)$ contains all literals with functional terms.

It is easily checked that this is a valid selection function for guarded clauses. If C is a non-ground clause, then it has at least one guard. Because this guard is negative, it is possible to select it. If C contains functional terms, then some of the literals containing functional terms are $\succ$-maximal. Because of this it is possible to select these literals.

The formula $\exists x \exists y\,(R_1(x,y) \wedge R_2(x,y) \wedge \forall z\,(R_1(y,z) \rightarrow R_2(y,z) \rightarrow P(z)))$, which is a translation of $\langle r_1 \wedge r_2 \rangle [r_1 \wedge r_2]p$ results in the following formula.

$$\begin{aligned}&\exists x \exists y\,(R_1(x,y) \wedge R_2(x,y) \wedge Q_{[r_1 \wedge r_2]p}(y))\\ &\quad \wedge \forall y z\,((R_1(y,z) \wedge R_2(y,z)) \rightarrow (Q_{[r_1 \wedge r_2]p}(y) \rightarrow P(z))).\end{aligned}$$

The clausal normal form consists of the clauses

$$\begin{aligned}&R_1(a,b)^*\\ &R_2(a,b)^*\\ &Q_{[r_1 \wedge r_2]p}(b)^*\\ &\neg R_1(x,y)^* \vee \neg R_2(x,y)^* \vee \neg Q_{[r_1 \wedge r_2]p}(x) \vee P(y).\end{aligned}$$

The literals marked with * are the maximal literals. The restriction could be completed, as in the previous section, in order to obtain a more total ordering. The termination proof is analogous to the proof for DL-clauses. The main difficulty is to prove that the restriction preserves the guarded fragment. For this we refer to Ganzinger and de Nivelle [14]. Consequently:

Theorem 6.2 *Let L be a logic in-between K and $K_{(m)}(\cap, \cup, \smile)$. Let N be the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$, where φ is any modal formula in L. Then:*

1. *Any derivation from N terminates in double exponential time.*
2. *φ is unsatisfiable in L iff the saturation of N contains the empty clause.*

7 Selection-Based Resolution for $K_{(m)}(\cap, \cup, \smile)$

$K_{(m)}(\cap, \cup, \smile)$ and logics below it have the property that they can be decided by a refinement of resolution which is defined solely by a selection function of negative literals [29].

Here new names are introduced for all non-atomic subformulae of the translation of a modal formula, that is, we use Def_Λ where Λ is the subset of positions in φ' (the first-order translation) which correspond to non-atomic subformulae of φ (the original modal formula). Moreover Def_Λ introduces the same symbol for variant subformulae with the same polarity. Because, by assumption, φ is in negation normal form, all occurrences of non-atomic subformulae of φ' with one free variable have positive polarity. This means $\mathrm{Def}_\lambda(\varphi') = \mathrm{Def}^+_\lambda(\varphi')$ for the positions λ associated with these occurrences. But subformulae corresponding to relational formulae (subformulae with two free variables) can occur both positively and negatively. For these Def_Λ introduces one symbol for all variant occurrences of subformulae corresponding to non-atomic relational subformulae with positive polarity and a different symbol for all variant occurrences with negative polarity.

For example, Def_Λ will introduce for the subformulae of $[\alpha]\langle\alpha\rangle p$ with $\alpha = r_1 \wedge r_2$ the definitions (in addition to $\exists x\, Q_{[\alpha]\langle\alpha\rangle p}(x)$):

$$
\begin{aligned}
&\forall x\,(Q_{[\alpha]\langle\alpha\rangle p}(x) \to \forall y\,(Q^n_\alpha(x,y) \to Q_{\langle\alpha\rangle p}(y)))\\
&\forall x\,(Q_{\langle\alpha\rangle p}(x) \to \exists y\,(Q^p_\alpha(x,y) \wedge P(y)))\\
&\forall xy\,(Q^p_\alpha(x,y) \to (R_1(x,y) \wedge R_2(x,y)))\\
&\forall xy\,((R_1(x,y) \wedge R_2(x,y)) \to Q^n_\alpha(x,y)).
\end{aligned}
\tag{7.1}
$$

The symbol Q^n_α (resp. Q^p_α) is associated with the negative (resp. positive) occurrence of α.

In order to characterise the induced class of clauses we introduce some more notation. We denote introduced predicate symbols by Q_ψ and Q^p_α or Q^n_α, where Q_ψ represents an occurrence of a modal subformula ψ and $Q^{p/n}_\alpha$ represents a positive/negative occurrence of a relational subformula α. We also find it convenient to use the notation

$$
\begin{aligned}
\mathcal{P}(s) &\quad \text{for some literal in } \{P_i(s), Q_\psi(s)\}_{i,\psi}, \text{ and}\\
\mathcal{R}(s,t) &\quad \text{for some literal in } \{R_j(s,t), R_j(t,s), Q^{p/n}_\alpha(s,t), Q^{p/n}_\alpha(t,s)\}_{j,\alpha}.
\end{aligned}
$$

Note the order of the arguments in $\mathcal{R}(s,t)$ is not fixed. Two occurrences of $\mathcal{P}(s)$, or $\mathcal{R}(s,t)$, need not be identical. For example, $\neg Q_\psi(x) \vee P_i(x) \vee Q_\chi(x)$ is an instance of $\neg Q_\psi(x) \vee \mathcal{P}(x) \vee \mathcal{P}(x)$, while

$$\neg Q_\psi(x) \vee \neg R_j(y,x) \vee Q_\chi(y) \quad \text{and} \quad \neg Q_\psi(x) \vee \neg Q^n_\alpha(x,y) \vee Q_\chi(y)$$

are instances of $\neg Q_\psi(x) \vee \neg \mathcal{R}(x,y) \vee \mathcal{P}(y)$.

Thus, all input clauses have one of the following forms.

$$
\begin{array}{ll}
\mathcal{P}(a) & \\
\neg Q_\psi(x)^* \vee \neg P_i(x)^* & \text{if } \psi = \neg p_i \\
\neg Q_\psi(x)^* \vee \mathcal{P}(x)\ [\vee\ \mathcal{P}(x)] & \text{if } \psi = \phi_1 \wedge[\vee]\ \phi_2 \\
\neg Q_\psi(x)^* \vee \neg \mathcal{R}(x,y)^*\ [\vee\ \mathcal{P}(y)] & \text{if } \psi = [\alpha]\phi\ [\psi = [\alpha]\bot] \\
\neg Q_\psi(x)^* \vee \mathcal{P}(f(x)) & \text{if } \psi = \langle\alpha\rangle\phi \\
\neg Q_\psi(x)^* \vee \mathcal{R}(x, f(x)) & \\
\neg Q^p_\alpha(x,y)^* \vee \mathcal{R}(x,y)\ [\vee\ \mathcal{R}(x,y)] & \text{if } \alpha = \beta_1 \wedge[\vee]\ \beta_2 \text{ has pos. polarity} \\
Q^n_\alpha(x,y) \vee \neg \mathcal{R}(x,y)^*\ [\vee\ \neg \mathcal{R}(x,y)^*] & \text{if } \alpha = \beta_1 \wedge[\vee]\ \beta_2 \text{ has neg. polarity.}
\end{array}
\tag{7.2}
$$

The literals marked with * are the selected literals.

The minimal calculus which we will use is based on maximal selection of negative literals. This means the selection function selects exactly the set of all negative literals in any non-positive clause. An ordering refinement is optional. The resolution rule is the following:

Resolution with maximal selection:

$$\frac{C_1 \vee A_1 \quad \ldots \quad C_n \vee A_n \quad \neg A_{n+1} \vee \ldots \vee \neg A_{2n} \vee D}{(C_1 \vee \ldots \vee C_n \vee D)\sigma}$$

provided for any $1 \leq i \leq n$, (i) σ is the most general unifier of A_i and A_{n+i}, (ii) $C_i \vee A_i$ and D are positive clauses, (iii) no A_i occurs in C_i, and (iv) $A_i, \neg A_{n+i}$ are selected.

The *negative premise* is $\neg A_{n+1} \vee \ldots \vee \neg A_{2n} \vee D$ and the other premises are the *positive premises*. The literals A_i and A_{n+i} are the *eligible literals*.

The inference rules of our calculus, denoted by $\mathsf{R}^{\mathcal{MOD}}$, are the above resolution rule, positive factoring, splitting and at least tautology deletion. All derivations in $\mathsf{R}^{\mathcal{MOD}}$ are generated by strategies in which no application of the resolution or factoring with identical premises and identical consequence may occur twice on the same path in any derivation. In addition, deletion rules, splitting, and the deduction rules are applied in this order, except that splitting is not applied to clauses which contain a selected literal.

As all non-unit clauses of a typical input set (a concrete example is given in Figure 2 below) contain a selected literal no factoring steps are possible and all definitional clauses can only be used as negative premises of resolution steps. To begin with there is only one candidate for a positive premise, namely, the ground unit clause $Q_\varphi(a)$ representing the input formula φ. Inferences with such ground unary unit clauses produce ground clauses consisting of positive literals only, which will be split into ground unit clauses.

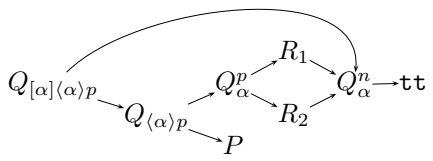


FIG. 1. Dependency among predicate symbols for (7.1)

Lemma 7.1 *Maximally split (non-empty) inferred clauses have one of two forms: $\mathcal{P}(s)$, or $\mathcal{R}(s, f(s))$, where s is a ground term.*

PROOF. Every resolution inference step with a definitional clause from the input set and ground unit clauses of the form $\mathcal{P}(s)$ or $\mathcal{R}(s, f(s))$ yields a ground clause which can be split into ground unit clauses of the required form. ∎

In general, s will be a nested non-constant functional ground term, which is usually undesirable, because in most situations this causes unbounded computations. However, as the next theorem proves, for the class of clauses under consideration any derived clause is smaller than its positive parent clauses with respect to a well-founded ordering which reflects the structure of the formula.

By definition the *modal depth* of a formula φ is the maximal nesting of modal operators $\langle\alpha\rangle$ or $[\alpha]$ in φ.

Theorem 7.2 *Let φ be any $K_{(m)}(\cap, \cup, \smile)$-formula and let N be the clausal form of* $\mathrm{Def}_\Lambda\Pi(\varphi)$. *Then:*

1. *Any $\mathsf{R}^{\mathcal{MOD}}$-derivation from N terminates.*
2. *φ is unsatisfiable in $K_{(m)}(\cap, \cup, \smile)$ iff the $\mathsf{R}^{\mathcal{MOD}}$-saturation of N contains the empty clause.*

PROOF. 2. follows from the soundness and refutational completeness of ordered resolution with selection (Theorem 4.1).

For 1., define a dependency relation $\succ_d$ on the predicate symbols by $S_1 \succ_d S_2$, if there is a definition $\psi \to \phi$ in $\mathrm{Def}_\Lambda\Pi(\varphi)$ such that S_1 occurs in ψ and S_2 occurs in ϕ. An additional restriction is that if Q_ψ is the symbol introduced for a diamond formula ψ, and Q_ϕ is the symbol introduced for a box formula ϕ, and ψ and ϕ occur at the same modal depth in φ, then $Q_\psi \succ_d Q_\phi$. Moreover, let $\mathtt{tt}$ be a new symbol smaller than all predicate symbols. (For example, for (7.1) the dependency relation is depicted in Figure 1. That is, $Q_{[\alpha]\langle\alpha\rangle p} \succ_d Q_{\langle\alpha\rangle p}$, and so on.) Let $\succ_D$ be any ordering on the predicate symbols in $\mathrm{Def}_\Lambda\Pi(\varphi)$ which is compatible with the transitive closure of $\succ_d$, that is, $\succ_d^+ \subseteq \succ_D$. Such an ordering can always be found. For this, it was important to introduce different predicate symbols for positive and negative subformulae associated with relational subformulae.

By definition, a predicate symbol Q is *associated with a function symbol f*, written Q_f, if there is a clause $\neg Q(x) \vee \mathcal{R}(x, f(x))$ in N. Define a measure μ as follows:

$$\mu(C) = \begin{cases} (\mathcal{P}, \mathcal{P}) & \text{if } C = \mathcal{P}(s) \\ (Q, \mathcal{R}) & \text{if } C = \mathcal{R}(s, t) \\ (\mathtt{tt}, \mathtt{tt}) & \text{if } C = \emptyset, \end{cases}$$

where Q is the predicate symbol associated with the leading function symbol of the maximal term in $\{s,t\}$. For example, the measure of a clause $R(s,f(s))$ is (Q_f,R). Complexity measures are compared by the lexicographic combination $\succ_c = (\succ_D,\succ_D)$. Now, it is routine to verify that any inference step from positive premises C_1,C_2 by resolution or factoring will produce a clause D such that $\mu(C_1) \succ_c \mu(D), \mu(C_2) \succ_c \mu(D)$. For example, for the inference step

$$\frac{Q_\psi(s)^+ \qquad \mathcal{R}(s,f(s))^+ \qquad \neg Q_\psi(x)^+ \vee \neg\mathcal{R}(x,y)^+ \vee \mathcal{P}(y)}{\mathcal{P}(f(s))}$$

$\mu(\mathcal{P}(f(s))) = (\mathcal{P},\mathcal{P})$, $\mu(Q_\psi(s)) = (Q_\psi,Q_\psi)$, and $\mu(\mathcal{R}(s,f(s))) = (Q_f,\mathcal{R})$, where Q_f is the symbol introduced for a diamond formula ϕ, say. ϕ cannot occur at a higher modal depth than ψ, which is a box formula. Hence, it follows that $Q_f(=Q_\phi) \succ_D \mathcal{P}$. Consequently, $\mu(\mathcal{R}(s,f(s))) \succ_c \mu(\mathcal{P}(f(s)))$. Since $Q_\psi \succ_D \mathcal{P}$, by definition, we also have that $\mu(Q_\psi(s)) \succ_c \mu(\mathcal{P}(f(s)))$. For

$$\frac{Q_\psi(s)^+ \qquad \neg Q_\psi(x)^+ \vee \mathcal{R}(x,f(x))}{\mathcal{R}(s,f(s))}$$

we have that $\mu(Q_\psi(s)) = (Q_\psi,Q_\psi) \succ_c (Q_\psi,\mathcal{R}) = \mu(\mathcal{R}(s,f(s)))$, because $Q_\psi \succ_D \mathcal{R}$. For the following inference, we have that $\mu(\mathcal{R}(s,f(s))) = (Q_f,\mathcal{R}) \succ_c (Q_f,Q^n_\alpha) = \mu(Q^n_\alpha(s,f(s)))$, since α occurs negatively in φ and thus $\mathcal{R} \succ_D Q^n_\alpha$.

$$\frac{\mathcal{R}(s,f(s))^+ \qquad Q^n_\alpha(x,y) \vee \neg\mathcal{R}(x,y)^+}{Q^n_\alpha(s,f(s))}$$

It follows that any derivation terminates. ∎

Theorem 7.3 *For any logic in-between K and $K_{(m)}(\cap,\cup,\smile)$, the space complexity for testing the satisfiability of a modal formulae φ with $\mathsf{R}^{\mathcal{MOD}}$ is bounded by $O(nd^m)$, where n is the number of symbols in φ, d is the number of different diamond subformulae in φ, and m is the modal depth of φ.*

Proof. Suppose φ is an arbitrary formula of $K_{(m)}(\cap,\cup,\smile)$ and N is the associated input set. φ has at most n subformulae, and hence, the number of clauses belonging to N is $O(n)$. Also, N contains at most n different predicate symbols (roughly one for each subformula), d different unary function symbols and one constant symbol. Recall from Lemma 7.1, split derived clauses are ground unit clauses of a certain form. As the maximal term depth bound is given by the modal depth of the input formula, there are at most $O(nd^m)$ such split clauses. It follows that the number of different literals in any derivation tree is bound by $O(nd^m)$. ∎

For logics without converse, space can be conserved by adopting the common tableaux inference strategy of considering disjunctive branches and branches associated with different $\Diamond$-subformulae in turn. In addition, the inferences with definitional clauses associated with diamond subformulae need to be postponed until no other inferences with definitional clauses associated with Boolean subformulae[3] are possible. This provides a PSPACE resolution procedure for logics in-between K and $K_{(m)}(\cap,\cup)$.

[3] In Boolean subformulae the outermost connective is a Boolean connective.

Automatically Generating Models

Any saturated clause set derivable from a given set N allows for the effective construction of a model of N. In general this model will not be finite. However, for $K_{(m)}(\cap, \cup, \smile)$ models are given by a finite set of positive ground unit clauses. The proofs of the results in this subsection are slight modifications of the corresponding results in Hustadt and Schmidt [29].

Formally, a *model* of a clause set is a set I of ground atoms. The presence of an atom A in I means A is true in I, and the absence of A means $\neg A$ is true in I. In general, a clause C is true in I iff for all ground substitutions σ there is a literal L in $C\sigma$ which is true in I. Falsehood is defined dually.

Lemma 7.4 *Let φ be a $K_{(m)}(\cap, \cup, \smile)$-formula. Let N be the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$, and let N_∞ denote the $\mathsf{R}^{\mathcal{MOD}}$-saturated clause set derivable from N. Let I be the set of positive ground unit clauses in N_∞. If N_∞ does not contain the empty clause then I is a model of N_∞ and N.*

Now it is an easy matter to construct a modal model $\mathcal{M} = (W, R, \iota)$ for φ from I. Essentially, the set of worlds is defined by the set of ground terms occurring in I. The interpretation of relational formulae is determined by the set of R_i literals in I. For any R_i, if $R_i(s,t)$ is in I then $(s,t) \in R(r_i)$, which can be extended to a homomorphism for complex relational formulae. The interpretation of modal formulae can be defined similarly. For any unary literal $P_i(s)$ (resp. $Q_\psi(s)$) in I, $s \in \iota(p_i)$ (resp. $s \in \iota(\psi)$), that is, p_i (resp. ψ) is true in the world s. This is homomorphically extended as expected. Consequently:

Theorem 7.5 *For any modal formula satisfiable in $K_{(m)}(\cap, \cup, \smile)$ a finite modal model can be effectively constructed on the basis of $\mathsf{R}^{\mathcal{MOD}}$.*

Corollary 7.6 *Let L be any logic in-between K and $K_{(m)}(\cap, \cup, \smile)$. Then, L has the finite model property.*

Generalisation

Results 7.2, 7.5 and 7.6 can be generalised.

Theorem 7.7 *Let L be a logic in-between K and $K_{(m)}(\cap, \cup, \smile)$. Let Δ be a finite $\mathsf{R}^{\mathcal{MOD}}$-saturated set of clauses consisting of two kinds of split components.*

(7.3) *Clauses with at most two free variables, which are built from finitely many binary predicate symbols R_j, no function symbols, and containing at least one guard literal (that is, this literal is negative and includes all the variables of the clause).*

(7.4) *Clauses built from one variable, finitely many function symbols (including constants), and finitely many binary predicate symbols R_j, with the restriction that (i) the argument multisets of all non-ground literals coincide, and (ii) each literal which contains a constant is ground.*

Suppose φ is an L-formula and N is the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$. Then:

1. *Any $\mathsf{R}^{\mathcal{MOD}}$-derivation from $N \cup \Delta$ terminates.*

2. φ is unsatisfiable in $L\Delta$ iff the $R^{\mathcal{MOD}}$-saturation of $N \cup \Delta$ contains the empty clause.

PROOF. Soundness and completeness follows by the general soundness and completeness result of ordered resolution with selection (Theorem 4.1).

For the problem of termination we first consider what kind of clauses we are dealing with. Input clauses have the form (7.2), (7.3) or (7.4). To begin with we consider the saturation of all theory clauses and the subset of clauses in N which contain only binary predicate symbols. The latter have the form:

$(')$ $\quad \neg Q^p_\alpha(x,y)^+ \vee \mathcal{R}(x,y)\ [\vee\ \mathcal{R}(x,y)] \quad$ and $\quad Q^n_\alpha(x,y) \vee \neg\mathcal{R}(x,y)^+\ [\vee\ \neg\mathcal{R}(x,y)^+]$.

We will prove that (non-empty maximally split) inferred clauses include ground unit clauses (more precisely, clauses of the form $[\neg]\mathcal{R}(s,t)$, where s and t are ground terms), and clauses which are specified by (7.4), except that they may be defined over the given R_j symbols and the introduced $Q^{p/n}_\alpha$ symbols. Call such clauses $(7.4)^+$. Let $\mathcal{K}$ denote the class of clauses $(')$, (7.3), $(7.4)^+$ and ground unit clauses, defined over a finite signature. W.l.o.g. we consider only maximally split clauses.

Claim 1: Inferences with clauses satisfying (7.3) and clauses $(')$ or ground unit clauses produce ground clauses only. Assume C is a (7.3) clause. C participates in inference steps as a negative premise and the only potential partners are ground clauses. As at least one of the eligible literals in C is a guard literal the result of such an inference step is a ground resolvent.

Claim 2: Inferences with clauses satisfying (7.3) and $(7.4)^+$ produce clauses with ground or $(7.4)^+$ split components. The proof is not difficult. Observe that the conclusion of a resolution step in $R^{\mathcal{MOD}}$ is always a positive clause.

Claim 3: Inferences with $(7.4)^+$ clauses and $(')$ clauses or ground unit clauses produce either ground clauses or $(7.4)^+$ clauses. First consider any positive $(7.4)^+$ clause C. Clearly, the factor of C is again a clause satisfying $(7.4)^+$. Now consider the possibilities for resolution inferences with C.

1. Assume C is resolved with a clause of the form $\neg Q_\psi(x)^+ \vee \neg\mathcal{R}(x,y)^+ \vee \mathcal{P}(y)$. The other positive premise besides C will be a ground unit clause $Q_\psi(s)^+$. Regardless of whether C is ground or not the conclusion will be a ground clause (because the single variable that may occur in C will be instantiated with a ground term).

2. Another possibility is that C is resolved with a clause $Q^n_\alpha(x,y) \vee \neg\mathcal{R}(x,y)^+$, or a clause $Q^n_\alpha(x,y) \vee \neg\mathcal{R}(x,y)^+ \vee \neg\mathcal{R}'(x,y)^+$. In the first case the resolvent is a variation of C, namely C with the predicate symbol of the eligible literal replaced by Q^n_α and possibly the arguments exchanged. In the second case the form of the resolvent depends on the second positive premise. If the second premise is ground then the resolvent will also be ground (because if C is not ground then the single variable of C will be instantiated with a ground term). The second premise C' may be a $(7.4)^+$ clause which is not ground. In this case a resolution step is only possible if the multisets of arguments of the eligible literals of C and C' are identical. It follows that any resolvent satisfies the conditions of $(7.4)^+$. Notice that no term depth growth occurs.

3. The third possibility is that C is resolved with a clause (7.3). This possibility is covered by Claim 2.

Second, consider the case that C is a non-positive $(7.4)^+$ clause. C can only be a negative premise in a resolution inference step. The only resolution partners are

ground unit clauses and positive $(7.4)^+$ clauses. In the first case the conclusion is a ground clause, and in the latter case the conclusion is again a $(7.4)^+$ clause.

Claim 4: Inferences with input clauses ($'$) and ground unit clauses produce ground clauses. The argument is similar as for Lemma 7.4.

Claims 1 to 4 prove that the class $\mathcal{K}$ is closed under inferences in $R^{\mathcal{MOD}}$. Now the saturation Δ' of Δ and the set of ($'$) clauses in N is a subset of $\mathcal{K}$. Δ' is bounded because inferred clauses contain at most two variables and there is no increase of the term depth.

We now establish that, conclusions of further inferences are ground. No inferences are possible between theory clauses satisfying condition (7.3) and clauses in N. Inferences with clauses not in Δ' are with ground clauses, and produce ground clauses. Similarly, inferences with clauses $(7.4)^+$ and clauses not in Δ' are with ground clauses, and produce ground clauses. The remaining inferences are as in Lemma 7.1. It follows that non-empty split ground conclusions have the form

$$\mathcal{P}(s), \quad (\neg)\mathcal{R}(s, f(s)), \quad (\neg)\mathcal{R}(s, s) \quad \text{or} \quad (\neg)\mathcal{R}(b, c).$$

Termination of any derivation from $N \cup \Delta$ is now shown as follows. Let $\mathtt{T}$, $\mathtt{tt}$ be new symbols which do not occur in either N or Δ. Again, we use a dependency relation $\succ_d$ on the predicate symbols. It is defined almost as in the proof of Theorem 7.2, but with subtle differences: $S_1 \succ_d S_2$, if there is a definition $\psi \to \phi$ in $\mathrm{Def}_\Lambda\Pi(\varphi)$ such that S_1 occurs in ψ and S_2 occurs in ϕ. In addition, all relational symbols R_j which do not occur in N are smaller than any unary predicate symbols. Let $\mathtt{T}$ be the largest symbol with respect to $\succ_d$, and $\mathtt{tt}$ the smallest symbol. As before, let $\succ_D$ be any ordering compatible with the transitive closure of $\succ_d$. In addition, define

$$\Gamma(T) = \{(\neg)R(s, t) \mid s, t \in T \text{ and } R \text{ is a binary predicate symbol}\}.$$

Let N_C denote the set of clauses derived prior to C, and C itself. Now, define a measure on (a subset of) clauses in a derivation by:

$$\mu(C) = \begin{cases} (\mathcal{P}, \emptyset) & \text{if } C = \mathcal{P}(s) \\ (Q, \Gamma(\{s, t\}) \setminus N_C) & \text{if } C = (\neg)\mathcal{R}(s, t) \\ (\mathtt{tt}, \emptyset) & \text{if } C = \emptyset, \end{cases}$$

where Q is the predicate symbol associated with the leading function symbol of the maximal term in $\{s, t\}$, whenever such a symbol exists, and $\mathtt{T}$ otherwise. Here, maximality is with respect to the proper subterm ordering. The ordering on the complexity measures $\succ_c$ of positive premises and conclusions is defined to be the lexicographic combination of $\succ_D$ and the proper superset relationship. This ordering is well-founded. Now we need to verify that split ground conclusions are strictly smaller than their positive premises, which is routine. Termination follows. ∎

Theorem 7.8 *Let L and Δ be as in the previous theorem. For any modal formula satisfiable in $L\Delta$ a finite modal model can be effectively constructed on the basis of $R^{\mathcal{MOD}}$.*

PROOF. The construction of a modal model $\mathcal{M}$ is as above from the set of ground unit literals in the saturation of $N \cup \Delta$. It remains to consistently complete $\mathcal{M}$ in

accordance with the background theory Δ. This is always possible because Δ contains no clauses requiring the creation of new worlds. For example, if $\Delta = \{R_i(x, f(x))\}$ then add $(s, s) \in R(r_i)$ for each dead end world s. Only propositional literals will be true in s. ∎

Corollary 7.9 *Let L and Δ be as in the previous theorem. Then, $L\Delta$ has the finite model property.*

Which extended modal logics satisfy the conditions of Theorem 7.7? Relational frame properties which can be described by the above clausal form include reflexivity, irreflexivity, seriality, symmetry, inclusions among relations, for example, $R_1 \subseteq R_2$ or $R_1 \subseteq (R_2^{\smile} \cap R_3)$, as well as, for example,

$$\forall x \exists y\, \neg R(x, y), \quad \forall x \exists y\, (R(x, y) \vee R(y, x)), \quad \text{or} \quad \forall xy\, (R(x, y) \rightarrow R(x, x)).$$

Thus, familiar logics covered by the above results include *KT*, *KD*, *KB*, *KTB*, and *KDB*, but also the basic tense logic K_t. The results also cover a variety of description logics, for example, $\mathcal{ALC}$ endowed with role conjunction, role disjunction and inverse roles, acyclic TBox statements, and both concept and role ABox statements.

By refining $\mathsf{R}^{\mathcal{MOD}}$ with an ordering restriction which would prefer to resolve upon literals containing functional terms of the theory clauses in Δ we expect that the above decidability result can be improved considerably.

Finally, let us look at a sample derivation, in Figure 2, and make a few observations. R is assumed to be reflexive, for otherwise not many inference steps are possible. First, notice how the correspondence to modal subformulae is retained during inference in $\mathsf{R}^{\mathcal{MOD}}$. For example, 14.1.1 and 14.2.1 say that p and $\Diamond p$ are true in the initial world a, 14.2.2, 14.2.3 and 14.2.4 say that p and $p \vee \Diamond p$ are true in a successor world of a. Second, notice the similarity of this derivation to the derivation of a classical tableaux procedure. This connection will be formally discussed in the next section.

8 Tableaux Calculi

Selection refinements of resolution (and hyperresolution) are closely related to standard modal tableaux calculi and description logic systems [13, 27, 28, 29]. In this section, we exploit this connection and present tableaux calculi for the modal logic $K_{(m)}(\cap, \cup, \smile)$, and logics below it. These calculi resemble and enhance those commonly used in description logic systems [22, 21]. We also investigate the relationship between our selection-based resolution procedure and single-step prefixed tableaux calculi.

Tableaux Calculi for Subsystems of $K_{(m)}(\cap, \cup, \smile)$

A *tableaux* is a finitely branching tree whose nodes are sets of labelled formulae. Given that φ is a formula to be tested for satisfiability the root node is the set $\{a : \varphi\}$. Successor nodes are constructed in accordance with a set of expansion rules. A rule $\frac{X}{X_1 \mid \ldots \mid X_n}$ fires for a selected formula F in a node if F is an instance of the numerator X, or more generally, F together with other formulae in the node are instances of the formulae in X. n successor nodes are created which contain the formulae of the current node and the appropriate instances of X_i. It is assumed that no rule is

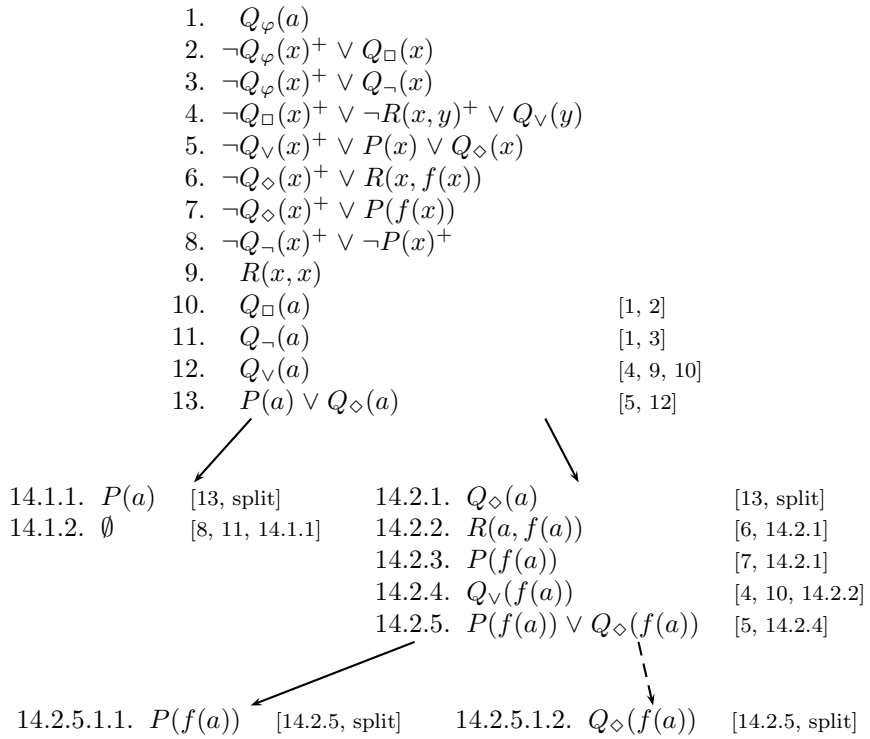


FIG. 2. Derivation tree for testing the satisfiability of $\varphi = \Box(p \vee \Diamond p) \wedge \neg p$ in KT.

applied twice to the same instance of the numerator. In the following we assume φ is a formula in negation normal form.

Figure 3 lists the expansion rules for the logic $K_{(m)}(\cap, \cup, \smile)$, while for any logic L in-between K and $K_{(m)}(\cap, \cup, \smile)$ the expansion rules are given by appropriate subsets, see Figure 4. The rules for $K_{(m)}(\cap, \cup, \smile)$ include the clash rule ($\perp$), seven 'elimination' rules ($\wedge$), ($\vee$), ($\Diamond$), ($\Box$), ($\smile$), ($\wedge^r$), and ($\vee^r$) for positive occurrences of subformulae, and three 'introduction' rules ($\smile_I$), ($\wedge^r_I$) and ($\vee^r_I$) for negative occurrences of subformulae. The side conditions for the introduction rules ensure that formulae are not introduced unnecessarily. Conjunction and disjunction are assumed to be associative and commutative operations. Note that only the disjunction rules are "don't know" nondeterministic and require the use of backtracking.

To avoid unnecessary duplication and superfluous inferences we define a notion of redundancy which is in the spirit of Bachmair and Ganzinger [3]. A labelled formula F is redundant in a node if the node contains labelled formulae $F_1, \ldots, F_n$ (for $n \geq 0$) which are smaller than F and $\models_L (F_1 \wedge \ldots \wedge F_n) \rightarrow F$. In this context a formula ψ is smaller than a formula ϕ if ψ is a subformula of ϕ, but a more general definition based on an admissible ordering in the sense of [3, 4] may be chosen. The application of a rule is redundant if its conclusion(s) is (are) redundant in the current node. For example, for any s, $s : \top$ is redundant, and if a node includes $s : \psi$ and $s : \psi \vee \phi$, then the ($\vee$) rule need not be applied, and no new branches are introduced.

$$(\bot)\ \frac{s:\psi,\ s:\neg\psi}{s:\bot} \qquad (\wedge)\ \frac{s:\psi\wedge\phi}{s:\psi,\ s:\phi} \qquad (\vee)\ \frac{s:\psi\vee\phi}{s:\psi \mid s:\phi}$$

$$(\Diamond)\ \frac{s:\langle\alpha\rangle\psi}{(s,t):\alpha,\ t:\psi}\ \text{with } t \text{ new to the branch} \qquad (\Box)\ \frac{(s,t):\alpha,\ s:[\alpha]\psi}{t:\psi}$$

$$(\smile)\ \frac{(s,t):\alpha^{\smile}}{(t,s):\alpha} \qquad (\wedge^r)\ \frac{(s,t):\alpha\wedge\beta}{(s,t):\alpha,\ (s,t):\beta} \qquad (\vee^r)\ \frac{(s,t):\alpha\vee\beta}{(s,t):\alpha \mid (s,t):\beta}$$

$$(\smile_I)\ \frac{(t,s):\alpha}{(s,t):\alpha^{\smile}} \qquad (\wedge^r_I)\ \frac{(s,t):\alpha,\ (s,t):\beta}{(s,t):\alpha\wedge\beta} \qquad (\vee^r_I)\ \frac{(s,t):\alpha}{(s,t):\alpha\vee\beta}$$

For the rules $(\smile_I)$, $(\wedge^r_I)$ and $(\vee^r_I)$ the side conditions are that the formulae in the denumerator, i.e. $\alpha^{\smile}$, $\alpha\wedge\beta$ or $\alpha\vee\beta$, occur as subformulae of the parameter γ of a box formula $s:[\gamma]\psi$ on the current branch.

FIG. 3. Tableaux expansion rules for $K_{(m)}(\cap,\cup,\smile)$.

For $K_{(m)}$:	$(\bot),(\wedge),(\vee),(\Diamond),(\Box)$
For $K_{(m)}(\smile)$:	$(\bot),(\wedge),(\vee),(\Diamond),(\Box),(\smile),(\smile_I)$
For $K_{(m)}(\cap)$:	$(\bot),(\wedge),(\vee),(\Diamond),(\Box),(\wedge^r),(\wedge^r_I)$
For $K_{(m)}(\cup)$:	$(\bot),(\wedge),(\vee),(\Diamond),(\Box),(\vee^r),(\vee^r_I)$
For $K_{(m)}(\cap,\cup)$:	$(\bot),(\wedge),(\vee),(\Diamond),(\Box),(\wedge^r),(\wedge^r_I),(\vee^r),(\vee^r_I)$
$\vdots$	$\vdots$

FIG. 4. Tableaux calculi for logics in-between $K_{(m)}$ and $K_{(m)}(\cap,\cup,\smile)$.

Theorem 8.1 *A formula φ is satisfiable in $K_{(m)}(\cap,\cup,\smile)$ iff a tableaux containing a branch $\mathcal{B}$ can be constructed with the rules of Figure 3 such that $\mathcal{B}$ does not contain the falsum ($s:\bot$ for some s) and each rule application is redundant.*

PROOF. By soundness, completeness and termination of the selection refinement $R^{\mathcal{MOD}}$ (Theorem 7.2), and the observation that the tableaux rules are macro inference steps of $R^{\mathcal{MOD}}$ on the set

$$N = N' \cup \{\neg Q^p_\alpha(x,y)^+ \vee Q^n_\alpha(x,y) \mid \alpha \text{ is a non-atomic relational formula in } \varphi\},$$

where N' is the clausal form of $\mathrm{Def}_\Lambda\Pi(\varphi)$, and Λ is as defined at the beginning of the previous section. For this extended N the termination argument is the same as in Theorem 7.2.

Define a mapping h' from labelled formulae to ground unit clauses by (h' is in fact a bijection)

$$\begin{aligned} h'(s:\psi) &= h(\psi)(h(s)) \\ h'((s,t):\alpha) &= h(\alpha)(h(s),h(t)), \end{aligned}$$

where ψ denotes a modal formula, α a relational formula. h is defined by: $h(p_i)=P_i$, $h(r_j)=R_j$, $h(\psi)=Q_\psi$, $h(\alpha)=Q^p_\alpha$, $h(a)=a$, and $h(t)=f_{\langle\alpha\rangle\psi}(h(s))$ where $s:\langle\alpha\rangle\psi$ is the formula for which t was introduced and $f_{\langle\alpha\rangle\psi}$ is the Skolem function associated with $\langle\alpha\rangle\psi$.

The $\mathsf{R}^{\mathcal{MOD}}$-derivation corresponding to an application of the $(\Diamond)$-rule is: from $Q_{\langle\alpha\rangle\psi}(h(s))$, $\neg Q_{\langle\alpha\rangle\psi}(x)^+ \vee Q^p_\alpha(x, f(x))$ and $\neg Q_{\langle\alpha\rangle\psi}(x)^+ \vee Q_\psi(f(x))$, derive the units $Q^p_\alpha(h(s), f(h(s)))$ and $Q_\psi(f(h(s)))$ in two resolution steps. For $(\smile_I)$ the resolvent of $Q^n_\alpha(h(s), h(t))$ (or $Q^p_\alpha(h(s), h(t))$ and $\neg Q^p_\alpha(x, y)^+ \vee Q^n_\alpha(x, y)$) and $Q^n_{\alpha^\smile}(x, y) \vee \neg Q^n_\alpha(y, x)^+$, is $Q^n_{\alpha^\smile}(h(t), h(s))$. Similarly, for the other rules.

Apart from factoring there are no inference steps in $\mathsf{R}^{\mathcal{MOD}}$ which are not involved in some macro inference step. Due to the fact that all positive premises are ground and thus subject to the application of splitting, factoring is not needed for completeness, and is thus optional. ∎

Corollary 8.2 *The appropriate subsets (see Figure 4) of the rules from Figure 3 provide sound, complete and terminating tableaux calculi for logics in-between K and* $K_{(m)}(\cap, \cup, \smile)$.

An immediate consequence of Theorem 7.5 is:

Corollary 8.3 *If L is a logic in-between K and* $K_{(m)}(\cap, \cup, \smile)$ *and* φ *is satisfiable in L then a finite modal model can be effectively constructed on the basis of the appropriate tableaux calculus for L.*

Simulation of Single-Step Prefixed Tableaux

We distinguish between two notions of polynomial simulation (or p-simulation). By definition, a proof system $\mathcal{A}$ *p-simulates derivations* of a proof system $\mathcal{B}$ iff there is a function g, computable in polynomial time, which maps derivations in $\mathcal{B}$ for any given formula φ, to derivations in $\mathcal{A}$ for φ. We also say system $\mathcal{A}$ *p-simulates search* of a system $\mathcal{B}$ iff there is a polynomial function g such that for any formula φ, g maps derivations from φ in $\mathcal{A}$ to derivations from φ in $\mathcal{B}$. The first notion generalises the notion of p-simulation found in [6], who are only concerned with the p-simulation of proofs (that is, successful derivations leading to a proof). Simulation of search is a relationship in the opposite direction. It implies that $\mathcal{A}$ does not perform any inference steps for which no corresponding inference steps exist in $\mathcal{B}$. To show that $\mathcal{A}$ p-simulates proofs or derivations of $\mathcal{B}$ it is sufficient to prove that for every formula φ and every derivation D_2 of φ in $\mathcal{B}$, there exists a derivation D_1 of φ in $\mathcal{A}$ such that the number of applications of inference rules in D_1 is polynomially bounded by the number of applications of inference rules in D_2. This can be achieved by showing that there exists a number n such that each application of an inference rule in D_1 corresponds to at most n applications of inference rules in D_2. It follows that the length of D_2 is polynomially bounded by the length of D_1. We call this a *step-wise simulation* of $\mathcal{B}$ by $\mathcal{A}$. Note that a step-wise simulation is independent of whether the considered derivations are proofs or not.

The single-step prefixed tableaux calculi of Massacci [31, 33] for subsystems of *S5* are defined by Figures 5 and 6. (Remember $KT = KDT$, $S4 = KT4$, $KB4 = KB5$, $S5 = KTB4 = KDB4 = KT5$.) The basic entities are formulae labelled with prefixes. A labelled (prefixed) formula has the form $\sigma : \varphi$, where σ is a sequence of positive integers and φ is a modal formula. σ represents a world in which φ is true. Tableaux derivations have a tree structure and begin with the formula, $1 : \varphi$ in the root node. Successor nodes are then constructed by the application of expansion rules. The

$$(\bot)\ \frac{\sigma : \psi,\ \sigma : \neg\psi}{\sigma : \bot} \qquad (\wedge)\ \frac{\sigma : \psi \wedge \phi}{\sigma : \psi,\ \sigma : \phi} \qquad (\vee)\ \frac{\sigma : \psi \vee \phi}{\sigma : \psi \mid \sigma : \phi}$$

$$(\Diamond)\ \frac{\sigma : \Diamond\psi}{\sigma.n : \psi}\ \text{with } \sigma.n \text{ new to the current branch}$$

$$(\Box)\ \frac{\sigma : \Box\psi}{\sigma.n : \psi} \qquad (D)\ \frac{\sigma : \Box\psi}{\sigma : \Diamond\psi} \qquad (T)\ \frac{\sigma : \Box\psi}{\sigma : \psi}$$

$$(B)\ \frac{\sigma.n : \Box\psi}{\sigma : \psi} \qquad (4)\ \frac{\sigma : \Box\psi}{\sigma.n : \Box\psi} \qquad (4^r)\ \frac{\sigma.n : \Box\psi}{\sigma : \Box\psi}$$

$$(4^d)\ \frac{\sigma.n : \Box\psi}{\sigma.n.m : \Box\psi} \qquad (5)\ \frac{1.n : \Box\psi}{1 : \Box\Box\psi}$$

FIG. 5. Single step prefixed tableaux expansion rules for subsystems of *S5*.

For K:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box)$
For KD:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (D)$
For KT:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (T)$
For KB:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (B)$
For $K4$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (4)$
For $K5$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (4^r), (4^d), (5)$
For KDB:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (D), (B)$
For $KD4$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (D), (4)$
For $KD5$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (D), (4^r), (4^d), (5)$
For KTB:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (T), (B)$
For $S4$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (T), (4)$
For $KB4$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (B), (4), (4^r)$
For $K45$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (4), (4^r), (4^d)$
For $KD45$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (D), (4), (4^r), (4^d)$
For $S5$:	$(\bot), (\wedge), (\vee), (\Diamond), (\Box), (T), (4), (4^r)$

FIG. 6. Tableaux calculi for subsystems of *S5*.

prefixes in the expansion rules, except for $\sigma.n$ of the $(\Diamond)$-rule, are assumed to be present on the current branch.

Theorem 8.4 (Massacci [31, 33], Goré [19]) *Let $\Sigma \subseteq \{D, T, B, 4, 5\}$. A formula φ is satisfiable in a logic $K\Sigma$ iff a tableaux containing a branch $\mathcal{B}$ can be constructed by the tableaux calculus for $K\Sigma$ such that $\mathcal{B}$ does not contain the falsum and further rule applications are redundant.*

The first-order background theories for the different axiom schemas are determined

by the following.

$$T_K = \emptyset \quad T_{KD} = \{R(x, f(x))^+\} \quad T_{KT} = \{R(x,x)^+\}$$
$$T_{KB} = \{\neg R(x,y)^+ \vee R(y,x)\} \qquad T_{K4} = \{\neg R(x,y)^+ \vee \neg R(y,z) \vee R(x,z)\}$$
$$T_{K5} = \{\neg R(x,y) \vee \neg R(x,z) \vee R(y,z), \neg R(x,y) \vee R(y,y)\}$$

For modal logics closed under more than one additional axiom schema the background theories are defined by the union of the corresponding clause sets, for example, $T_{KD4} = T_{KD} \cup T_{K4}$.

Observe that for 4 and 5 only certain negative literals will be selected in the theory clauses. In the case of 5 we do not select any literal.

Theorem 8.5 *Let $\Sigma \subseteq \{D, T, B, 4, 5\}$. Resolution p-simulates derivations of single step prefix tableaux for $K\Sigma$.*

PROOF. Suppose we are interested in the satisfiability of the modal formula φ. We will show that $\mathsf{R}^{\mathcal{MOD}}$ p-simulates single step prefix tableaux step-wise.

Similar as in the proof of Theorem 8.1 define a mapping (bijection) h' from prefixed formulae to ground unit clauses by $h'(\sigma : \psi) = h(\psi)(h(\sigma))$, where h is defined by: $h(p_i) = P_i$, $h(r_j) = R_j$, $h(\psi) = Q_\psi$ for ψ a modal subformula of φ, $h(1) = a$, and $h(\sigma.n) = f_{\Diamond\psi}(h(\sigma))$ where $\Diamond\psi$ is the formula for which n was introduced and $f_{\Diamond\psi}$ is the Skolem function associated with $\Diamond\psi$. For example, the unit clause associated (by h') with the formula $1 : \varphi$ contained in the root node is $Q_\varphi(a)$.

Now show that each tableaux inference step can be simulated by a constant number of $\mathsf{R}^{\mathcal{MOD}}$-inference steps. For instance, the derivation of $\bot$ by the clash rule corresponds to one resolution inference step applied to $Q_\psi(h(\sigma))$, $Q_{\neg\psi}(h(\sigma))$ and $\neg Q_{\neg\psi}(x)^+ \vee \neg Q_\psi(x)^+$, which generates the empty clause. For the simulation of the application of the $(\Diamond)$ rule to $\sigma : \Diamond\psi$ we may assume that $Q_{\Diamond\psi}(h(\sigma))$ is present in the clauses set. Also present are the definitional clauses $\neg Q_{\Diamond\psi}(x)^+ \vee R(x, f(x))$, and $\neg Q_{\Diamond\psi}(x)^+ \vee Q_\psi(f(x))$. Then an application of the $(\Diamond)$ rule corresponds to performing two resolution inference steps producing $R(h(\sigma), f(h(\sigma)))$ and $Q_\psi(f(h(\sigma)))$. The term $f(h(\sigma))$ corresponds to the new prefix $\sigma.n$. The interested reader may fill in the details for the other rules, see also [29]. ∎

For the modal logics $K\Sigma$ with $\Sigma \subseteq \{D, T, B\}$ there is a near bisimulation between the tableaux calculi and $\mathsf{R}^{\mathcal{MOD}}$. If factoring rules are added to the tableaux calculi then tableaux p-simulates also derivations of the selection-based resolution refinement. It follows that:

Theorem 8.6 *$\mathsf{R}^{\mathcal{MOD}}$ p-simulates search in single step prefix tableaux for $K\Sigma$ with $\Sigma \subseteq \{D, T, B\}$.*

This is not true for logics in which 4 and 5 are theorems. For 4 and 5 termination in single step prefixed tableaux is ensured by a loop checking mechanism [31, 33]. Once a loop is detected in a branch no further rules are applied. In $\mathsf{R}^{\mathcal{MOD}}$ further inference steps will be performed. To prevent this we have to provide the means by which the resolution procedure can recognise the redundancy of further inference steps. This may possibly be realised by soft typing [16] or some form of blocking which is analogous to loop checking [27].

In this section we have focussed on single-step prefixed tableaux calculi, but this choice is arbitrary. Our technique can also be applied for obtaining simulation results of modal tableaux calculi with implicit or explicit accessibility relation and analytic modal KE tableaux [25, 32], or even sequent proof systems. Simulation results of tableaux calculi for description logics by resolution can be found in Hustadt and Schmidt [27, 28].

9 Concluding Remarks

The approach purported in this overview paper is that modal logics can be seen to be fragments of first-order logic and inference systems for modal logics can be developed and studied within the framework of first-order resolution. Several issues were considered. In particular, we have focussed on the decision problem for a range of expressive extended modal logics and have described resolution procedures of varying nature. We have looked at using resolution methods for automatically generating models. Exploiting the link between selection-based resolution and tableaux methods, we have proposed a new tableaux calculus for multi-modal logics defined over relations closed under union, intersection and converse. And, we have presented simulation results which give us an understanding of modal tableaux methods in the wider context of first-order logic and resolution.

Some important modal logics for which we have not presented a decision procedure are modal logics with transitive modalities. To decide extensions of *K4* one possibility is to modify the calculus and add ordered chaining rules for transitive relations [15]. Another possibility is to use the resolution procedures described in this paper but block further inferences with clauses containing terms in which the level of nesting exceeds a pre-computed term depth bound. In practice this solution is rather poor, as are solutions encoding *K4* or *S4* problems in *K* or *KT*.

Acknowledgements

We thank the referees for valuable and detailed comments.

References

[1] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. of Philosophical Logic*, 27(3):217–274, 1998.

[2] H. Andréka, J. van Benthem, and I. Németi. Back and forth between modal logic and classical logic. *Bull. IGPL*, 3(5):685–720, 1995.

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

[4] L. Bachmair and H. Ganzinger. Resolution theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 2000. To appear.

[5] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics Series. Academic Press, New York, 1973.

[6] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *J. of Symbolic Logic*, 44(1):36–50, 1979.

[7] H. de Nivelle. *Ordering Refinements of Resolution*. PhD thesis, Delft University of Technology, 1995.

[8] H. de Nivelle. An overview of resolution decision procedures. In M. Faller and S. Kaufmann, editors, *Formalizing the Dynamics of Information*. 1998.

[9] H. de Nivelle. A resolution decision procedure for the guarded fragment. In C. Kirchner and H. Kirchner, editors, *Automated Deduction—CADE-15*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 191–204. Springer, 1998.

[10] H. de Nivelle. Bliksem, 1999. `http://www.mpi-sb.mpg.de/~bliksem`.

[11] M. de Rijke. A system of dynamic modal logic. *J. of Philosophical Logic*, pages 109–142, 1998.

[12] C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Method for the Decision Problem*, volume 679 of *Lecture Notes in Computer Science*. Springer, 1993.

[13] C. G. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 2000. To appear.

[14] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 295–303. IEEE Computer Society Press, 1999.

[15] H. Ganzinger, U. Hustadt, C. Meyer, and R. A. Schmidt. A resolution-based decision procedure for extensions of K4. In K. Segerberg, M. Zakharyaschev, M. de Rijke, and H. Wansing, editors, *Advances in Modal Logic, Volume 2*, Lecture Notes. CSLI Publications, Stanford, 1999. To appear.

[16] H. Ganzinger, C. Meyer, and C. Weidenbach. Soft typing for ordered resolution. In *Proceedings of the 14th International Conference on Automated Deduction, CADE-14*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 321–335. Springer, 1997.

[17] G. Gargov and S. Passy. A note on Boolean modal logic. In P. P. Petkov, editor, *Mathematical Logic: Proceedings of the 1988 Heyting Summerschool*, pages 299–309. Plenum Press, 1990.

[18] G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, editor, *Mathematical Logic and its Applications: Proceedings of the 1986 Gödel Conference*, pages 253–263. Plenum Press, 1987.

[19] R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, 1999.

[20] E. Grädel. On the restraining power of guards. To appear in the *J. of Symbolic Logic*, 1998.

[21] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. Research Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany, 1990.

[22] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647. Morgan Kaufmann, 1998.

[23] I. L. Humberstone. Inaccessible worlds. *Notre Dame J. of Formal Logic*, 24(3):346–352, 1983.

[24] I. L. Humberstone. The modal logic of 'all and only'. *Notre Dame J. of Formal Logic*, 28(2):177–188, 1987.

[25] U. Hustadt and R. A. Schmidt. Simplification and backjumping in modal tableau. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98, Oisterwijk, The Netherlands, Proceedings*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 187–201. Springer, 1998.

[26] U. Hustadt and R. A. Schmidt. Maslov's class K revisited. In H. Ganzinger, editor, *Automated Deduction—CADE-16*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 172–186. Springer, 1999.

[27] U. Hustadt and R. A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 110–115. Morgan Kaufmann, 1999.

[28] U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Artificial Intelligence*, pages 192–206. Springer, 2000.

[29] U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. To appear in the *SAT 2000* Special Issue of *J. of Automated Reasoning*, 2000.

[30] A. Leitsch. *The Resolution Calculus*. EATCS Texts in Theoretical Computer Science. Springer, 1997.

[31] F. Massacci. Strongly analytic tableaux for normal modal logics. In A. Bundy, editor, *Automated Deduction—CADE-12*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 723–737. Springer, 1994.

[32] F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98, Oisterwijk, The Netherlands, Proceedings*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 217–231. Springer, 1998.

[33] F. Massacci. Single step tableaux for modal logics: Computational properties, complexity and methodology. *J. of Automated Reasoning*, 2000. To appear.

[34] W. McCune. *Otter*, 1995. `http://www.mcs.anl.gov/AR/otter/`.

[35] H. J. Ohlbach. Translation methods for non-classical logics: An overview. *Bulletin of the IGPL*, 1(1):69–89, 1993.

[36] H. J. Ohlbach. Combining Hilbert style and semantic reasoning in a resolution framework. In C. Kirchner and H. Kirchner, editors, *Automated Deduction – CADE-15, 15th International Conference on Automated Deduction*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 205–219. Springer, 1998.

[37] H. J. Ohlbach, A. Nonnengart, and D. Gabbay. Encoding two-valued non-classical logics in classical logic. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 2000. To appear.

[38] S. Passy and T. Tinchev. PDL with data constants. *Information Processing Letters*, 20:35–41, 1985.

[39] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *J. of Symbolic Computation*, 2:293–304, 1986.

[40] C. Weidenbach, et al. SPASS, 1999. `http://spass.mpi-sb.mpg.de`.

Received February 14, 2000

An Analysis of Empirical Testing for Modal Decision Procedures

IAN HORROCKS, *Department of Computer Science, University of Manchester, UK. E-mail: horrocks@cs.man.ac.uk*

PETER F. PATEL-SCHNEIDER, *Bell Labs Research, Murray Hill, NJ, U.S.A. E-mail: pfps@research.bell-labs.com*

ROBERTO SEBASTIANI, *DISA - Università di Trento, Via Inama 5, I-38100, Trento, Italy. E-mail: rseba@cs.unitn.it*

Abstract

Recent years have seen the emergence of a new generation of heavily-optimised modal decision procedures. Several systems based on such procedures are now available and have proved to be much more effective than the previous generation of modal decision procedures. As both computational complexity and algorithm complexity are generally unchanged, neither is useful in analysing and comparing these new systems and their various optimisations. Instead, empirical testing has been widely used, both for comparison and as a tool for tuning systems and identifying their strengths and weaknesses. However, the very effectiveness of the new systems has revealed serious weaknesses in existing empirical test suites and methodologies. This paper provides a detailed survey of empirical testing methodologies, analyses the current state of the art and presents new results obtained with a recently developed test method.

Keywords: empirical testing, modal decision procedures

1 Motivations for Empirical Testing

Heavily-optimised systems for determining satisfiability of formulae in propositional modal logics are becoming available. These systems, including DLP [30], FaCT [23], KSatC [13], *SAT [35], and TA [26], have more optimisations and are much faster than the previous generation of modal decision procedures, such as leanK [3], Logics Workbench [21], and $\Box$**KE** [32].

As with most theorem proving problems, neither computational complexity nor algorithm complexity is useful in determining the effectiveness of optimisations. The worst-case complexity of the problem, of course, remains unchanged. For many propositional modal logics, this complexity ranges from PSPACE-complete to EXPTIME-complete. The worst-case complexity of the algorithms in the systems also generally remains unchanged under optimisation. The worst-case complexity for most of these systems is exponential time and either polynomial or exponential space. Further, determining any useful normal-case or special-case complexity is essentially impossible.

As theoretical studies do not provide any indication of the effectiveness of the new systems and their optimisations, this has to be determined by empirical testing. In any case, empirical testing provides a number of benefits over theoretical complexity.

It directly gives resource consumption, in terms of computation time and memory use. It factors in all the pieces of the system, not just the basic algorithm itself.

Empirical testing can be used not only to compare different systems, but also to tune a system with parameters that can be used to modify its performance. Moreover, it can be used to show what sort of inputs the system handles well, and what sort of inputs the system handles poorly.

In this paper we provide a detailed survey of empirical testing methodologies for modal decision procedures, including a review of previous work in the area and an analysis of the current state of the art.[1] We point out desirable and undesirable characteristics of these methodologies. We also present some new results obtained with a recently developed test method, and identify some of the remaining weaknesses in both modal decision procedures and testing methodologies.

Our goal in this paper is not to show the effectiveness of various systems and their optimisations, but is instead to provide a framework for evaluating empirical testing methodologies for modal decision procedures, to analyse these methodologies, and to give a direction for better empirical testing methodologies for modal decision procedures.

2 Evaluating Empirical Testing

2.1 Kinds of Empirical Test Sets

Several kinds of inputs can be used for empirical testing. Inputs that have been encountered in the past can be used. Variations on these past inputs, either systematic or random, can be used. It is also possible to deterministically synthesise inputs. These can be either hand-generated individual inputs, possibly parameterised, or inputs that systematically cover an area. Finally, randomly-generated inputs can be used. Each of these kinds of inputs has benefits and potential problems.

Actual past inputs provide a good mechanism for comparing the behaviour of different systems. However, as systems improve, old inputs can become so easy that they provide no guidance. Also, newer inputs will not necessarily be the same as past inputs, so using past inputs may not provide guidance for future performance. Further, there may not be enough past inputs to provide sufficient testing.

Variations on past inputs can be used to overcome some of these problems. Variations provide more inputs. If the past inputs can be modified to be larger or more difficult, then the problem of past inputs being too easy can be overcome.

Hand-generated inputs can be specifically tailored to provide good tests, at least for particular systems. In particular, if the inputs are parameterised, then they can often be made large enough to be difficult, even for newer systems. However, it can be very hard to hand-generate difficult problems, even if parameterised—a particular optimisation may make a whole parameterised set of inputs trivial. Further, hand-generation is expensive.

Systematic inputs can be very effective, provided that systematic generation is possible. However, many kinds of inputs are too large to systematically cover. Random inputs are often easy to generate, and can provide a mechanism to cover a class of inputs. Both systematic and random inputs may not be typical inputs, and so testing

[1] Some of the testing in this paper has been reported on in other papers [25; 24; 13].

using these kinds of inputs may not provide useful data.

In many cases there is a lack of past inputs, not enough resources for hand-generation, and no way of systematically covering the input space, so the major mechanism for generating inputs for empirical testing has to be random generation. This is the case in propositional modal logics. The current main empirical testing methodologies involve random generation of formulae. The only other significant test consists of hand generated, parameterised formulae, but this test has become too easy for state-of-the-art systems.

2.2 Good and Bad Empirical Testing

The benefits of empirical testing depend on the characteristics of the inputs provided for the testing, as empirical testing only provides data on these particular inputs. If the inputs are not typical or suitable, then the results of the empirical testing will not be useful. This means that the inputs for empirical testing must be carefully chosen.

We believe that good test sets should be created according to the following key criteria.[2]

Reproducibility. Any test is not very useful if it cannot be reproduced, and varied. The test formulae or their generation function should thus be made available. Even if the test formulae are made available, the generation function should also be made available, so that variants of the test can be developed. If the actual test formulae are not made available it should be possible to *exactly* reproduce the entire test set, so all the inputs to the generation function should be disclosed, including any "random" or environmental inputs.

Representativeness. The ideal test set should represent a significant area of the whole input space, and should span the whole range of sources of difficulty. A good empirical test set should at least cover a large area of inputs. Empirical test sets that consist of only a few inputs or that concentrate on only a small area of the input space provide no information about most inputs. This can be a particular problem if the small area has a different computational complexity than the input space as a whole.

Valid vs. not-valid balance. In a good test set, valid and not-valid (or, equivalently, satisfiable and unsatisfiable) problems should be more or less equal both in number and in difficulty. In fact, solvable and unsolvable problems may present different sources of difficulty, so that a system which is good at handling one type may be not good—or not capable at all—of handling the other type. Moreover, to prevent the usage of routines/heuristics which are explicitly aimed at detecting either solvability or unsolvability, the testbed should provide no *a priori* information which could help in guessing the result —that is, *maximum uncertainty* regarding the solvability of the problems is desirable. (Notice that, in the real world, the solvability of a problem is not known a priori.)

Difficulty. A good empirical test set should provide a sufficient level of difficulty for the system(s) being tested. (Some problems should be too hard even for state-of-the-art systems, so as to be a good benchmark for forthcoming systems.) If the

[2] Notice that some of the criteria are identical or similar to those suggested by Heurding and Schwendimann[22].

inputs are too easy, then the resulting resource consumption may be too small to easily measure, and the resource consumption may be dominated by start-up costs that do not grow as the difficulty of the inputs grow. Comparing absolute performances—which may depend on factors like the platform used, the quality of implementation, etc.—may be less significant than comparing how performances scale up with problems of increasing difficulty.

Termination. To be of practical use, the tests should terminate and provide information within a reasonable amount of time. If the inputs are too hard, then the system may not be able to provide answers within the established time. This inability of the system is of interest, but can make system comparison impossible or insignificant.

The following criteria derive from or are significant sub-cases of the main criteria above.

Parameterisation. One way of creating test sets with the appropriate features and with a large number of inputs is to have parameterised inputs with sufficient parameters and degrees of freedom to allow the inputs to range over a large portion of the input space. On the other hand, the number of parameters and their degrees of freedom should not be too large, otherwise the number of tests required to cover a significant subspace might blow up. (Ideally, the parameter set should work as much as possible as a "base" for the input space.)

Control. In particular, it is very useful to have parameters that control *monotonically* the key features of the input test set, like the average difficulty and the solvable vs. unsolvable rate. Monotonicity is a key point, as it allows for controlling one feature independently of the values of the other parameters, and for eliminating uninteresting areas of the input space.

Modal vs. propositional balance. Reasoning in modal logics involves alternating between two orthogonal search efforts: pure modal reasoning—that is, spanning the potential Kripke models—and pure propositional reasoning—that is, assigning truth values to sub-formulae within each Kripke state. A good test set should be challenging from both viewpoints.

Data organisation. The data should be summarisable—so as to make a comparison possible with a limited effort—and plottable—so as to enable the qualitative behaviour of the system(s) to be highlighted. For instance, a list of hundreds of uncorrelated numbers is not a well-organised data set, since it makes a comparison impractical, and makes it very hard to produce any qualitative information from it.

Focus on narrow problems. As an alternative to wide-ranging tests, small "ad hoc" test sets may be used for testing systems on one particular source of difficulty, or for revealing one particular possible weakness. For instance, formulae which are satisfied only by exponentially-large Kripke models (see, e.g., [20]) might cause the system under test to blow up in space, thus revealing its non-PSPACEness.

Finally, in creating good test sets, particular care must be taken to avoid the following problems.

Redundancy. Empirical test sets must be carefully chosen so as not to include inadvertent redundancy. They should also be chosen so as not to include small

sub-inputs that dictate the result of the entire input. Empirical test sets can be made irrelevant by advances in systems if the advanced systems include optimisations that identify some inherent redundancy and cause the test set to be trivially solved. Of course, a system that can detect such redundancy is better than one that cannot, but the presence of detectable redundancies can reduce test sets to triviality.

Triviality. A good test set should be flawless, that is, it should not contain significant subsets of trivial problems. This problem has claimed victims in many other areas of AI, as flaws have been detected in random test methods for propositional satisfiability [29], for constraint satisfiability problems [1], and for quantified boolean formulae [10].

Artificiality. A good empirical test set should correspond closely to inputs from applications. If the test set does not resemble actual inputs, then the results from the empirical testing will not necessarily correspond with the behaviour of the system in real use.

Over-size. The single problems should not be too big with respect to their difficulty, so that the resources required for parsing and data managing do not seriously influence total performance.

In general, these criteria boil down to providing a reproducible sample of an interesting portion of the input space with appropriate difficulty. This is no different from the criteria in other areas, notably propositional satisfiability testing [9; 7; 33], theorem proving [34], CSP [7; 11]. However, the situation for modal decision procedures is more difficult than for the fields above, because of the greater variety in modal logics and formulae and the lesser capabilities of current modal decision procedures.

3 Systems

The systems involved in most of this testing have different characteristics, but are all based around a front-end that takes an input formula, converts it into an internal form, and uses a search engine to exhaustively search for a model of the formula. The input formula is satisfiable if such a model is found and unsatisfiable otherwise. All the systems are able to handle (at least) the propositional modal logic $\mathbf{K_{(m)}}$.

Two systems, DLP [30] and FaCT [23], are based on custom-built search engines that employ tableaux techniques to search for the model. These systems both translate input formulae into an internal normal form, attempting to exploit redundancies and local analytic truth and falsity. Their search engines employ mechanisms to reduce overlapping search, cut off search branches that cannot succeed, detect forced branches, and reuse cached results from previous searching.

Both DLP and FaCT can handle logics that are supersets of $\mathbf{K_{(m)}}$. FaCT allows transitive modalities, deterministic (functional) modalities and inclusion relationships between modalities. DLP allows full propositional dynamic logic, although it has a compile-time switch to change from propositional dynamic logic to $\mathbf{S4_{(m)}}$. Most of the tests in this paper will use only DLP as it is based on the ideas developed in the FaCT system, includes most of FaCT's optimisations, and has some additional optimisations (in particular caching). If not otherwise specified, all the examples

with DLP were obtained with DLP version 3.1 in its $\mathbf{S4}_{(\mathbf{m})}$ configuration running on a machine roughly comparable to a SPARC Ultra 1 with 256MB of main memory.

Two other systems, KSatC [13] and *SAT [35], are based on state-of-the-art propositional satisfiability testing procedures. The two systems make multiple calls to the propositional decision procedure. While KSatC's optimisations are largely those of the underlying propositional system, *SAT features many modal search pruning optimisations like modal backjumping and caching. *SAT also handles many non-normal modal logics. If not otherwise specified, all the examples with *SAT in this paper were obtained with *SAT `-e -m6`, compiled with `Linux gcc -O2` and run on a 350MHz PentiumII with 128MB of main memory.

The TA system [26] translates propositional formulae into a decidable fragment of first order logic. It uses an optimised first-order theorem prover to determine the satisfiability of the translated formulae. TA thus inherits the optimisations built into this theorem prover, but it also uses a translation that is designed to produce easier first-order formulae.

4 The Heuerding and Schwendimann Tests

As discussed in Section 2.1, it is possible to hand-generate formulae for testing modal decision procedures. In the past such formulae were difficult to analyse, but the optimised decision procedures that are now available make short work of such hand-generated formulae. For example, Heuerding and Schwendimann [22] report that their (moderately-optimised) system, LWB, can rapidly process several previous collections, with the longest test taking under 1/10th of a second.

Such short times are not satisfactory as differences between systems may be the result of startup costs and not indicative of their behaviour on more-difficult formulae.

4.1 Rationale

To overcome the above difficulty, and also to provide more test formulae, Heuerding and Schwendimann [22] created a suite of formulae for testing modal decision procedures. They wanted to provide a test suite that would not be quickly rendered obsolete and that would provide a comprehensive test of a modal decision procedure, so they started with a number of postulates:[3]

1. The test suite should include valid as well as invalid formulae.
2. The test suite should provide formulae of various structures.
3. Some of the formulae should be hard for future systems.
4. The validity status of the formulae should be known in advance.
5. The formulae should be resistant to simple tricks.
6. Executing the entire benchmark should not take an excessive amount of time.
7. It should be possible to summarise succinctly the results of the benchmark.

[3] These postulates are elaborated in the paper by Heuerding and Schwendimann [22, pp. 2–3].

4.2 Description

To meet these postulates Heuerding and Schwendimann created classes of formulae. Each class was generated from a (relatively) simple parameterised logical formula that was either valid or invalid. Some of these formulae were made harder by hiding their structure or adding extra pieces. The parameters allow formulae of different size to be created, thus allowing for formulae of differing difficulty. The idea behind the parameter is that the difficulty of most of the problems should be exponential in the parameter. This supposed exponential increase in difficulty would make differences in the speed of the machines used to run the benchmarks relatively insignificant.

For each logic, **K**, **KT**, and **S4**, 9 classes of formula were created, in both valid and invalid versions. For example, the branching formulae of Halpern and Moses [20], form a formula class for all three logics. Other problem classes in the set are based on the pigeon-hole principle and a two-colouring problem on polygons.

The benchmark methodology was to test formulae from each class, starting with the easiest instance, until the validity status of a formula could not be correctly determined within 100 seconds. The result from this class would then be the parameter of the largest formula that could be solved within the time limit. The parameter ranges only from 1 to 21—if a system can solve all 21 instances of a class, the result is given as ">".

This benchmark suite and methodology meets several of the postulates above simply as a result of its design. The suite contains both valid and invalid formulae of various structures whose validity status is known in advance. The benchmark can be executed in a few hours at most and the results can be given in three tables each with nine rows and two columns.

4.3 Results

The benchmark suite was used in a comparison at Tableaux'98 [2]. Six entries were submitted to this comparison, giving results for a total of ten systems. Since then the benchmark has been run on several other systems, including some more-recent versions of systems included in the original test. Several results are given in Tables 1, 2, and 3. Some of these results are from the Tableaux'98 comparison, but some are more recent.

The results show that this benchmark is appropriate, perhaps even too difficult, for some of the systems. However, the heavily-optimised systems, including *SAT and DLP, are able to handle all of the instances of many of the problem classes in their areas of coverage. *SAT is able to completely solve 15 out of 18 of the **K** tests and DLP is able to completely solve 11 out of 18 of both the **KT** tests and the **S4** tests. This means that the effective number of tests is reduced considerably.

In fact, the situation is even worse than indicated by the raw results. The heavily-optimised systems can solve many of the problem classes with little or no search. This is indicated in Table 4, which gives the time taken for the most-difficult solved problems in **K** for *SAT, DLP and TA.[4] The times for TA are subdivided between FLOTTER (a pre-processor) and SPASS itself.

[4] These results differ slightly from the previous results for DLP and differ considerably for TA because they were performed on a different version of the systems.

	branch		*d4*		*dum*		*grz*		*lin*		*path*		*ph*		*poly*		*t4p*	
K	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*
leanK 2.0	1	0	1	1	0	0	0	>	>	4	2	0	3	1	2	0	0	0
□KE	13	3	13	3	4	4	3	1	>	2	17	5	4	3	17	0	0	3
LWB 1.0	6	7	8	6	13	19	7	13	11	8	12	10	4	8	8	11	8	7
TA	9	9	>	18	>	>	>	>	>	>	20	20	6	9	16	17	>	19
KSAT	8	8	8	5	11	>	17	>	>	3	4	8	5	5	13	12	10	18
***SAT 1.2**	>	12	>	>	>	>	>	>	>	>	>	>	8	12	>	>	>	>
Crack 1.0	2	1	2	3	3	>	1	>	5	2	2	6	2	3	>	>	1	1
Kris	3	3	8	6	15	>	13	>	6	9	3	11	4	5	11	>	7	5
FaCT 1.2	6	4	>	8	>	>	>	>	>	>	7	6	6	7	>	>	>	>
DLP 3.1	19	13	>	>	>	>	>	>	>	>	>	>	7	9	>	>	>	>

TABLE 1. Results for **K**

	45		*branch*		*dum*		*grz*		*md*		*path*		*ph*		*poly*		*t4p*	
K	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*
leanKT 2.0	3	0	1	0	3	4	0	0	3	2	2	1	2	1	0	0	>	0
□KE	14	2	16	15	1	1	0	>	4	4	16	6	4	3	0	0	7	17
LWB 1.0	5	4	5	6	5	10	6	>	5	5	10	9	4	8	14	2	5	7
TA	17	6	13	9	17	9	>	>	16	20	>	16	5	12	>	1	11	0
KSAT	5	5	8	7	7	12	9	>	2	4	2	5	4	5	1	2	1	1
Crack 1.0	0	0	2	2	0	1	0	0	2	4	1	5	2	2	1	1	0	1
Kris	4	3	3	3	3	14	0	5	3	4	1	13	3	3	2	2	1	7
FaCT 1.2	>	>	6	4	11	>	>	>	4	5	5	3	6	7	>	7	4	2
DLP 3.1	>	>	19	12	>	>	>	>	3	>	16	14	7	>	>	12	>	>

TABLE 2. Results for **KT**

	45		*branch*		*dum*		*grz*		*md*		*path*		*ph*		*poly*		*t4p*	
K	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*
KT4	1	6	2	3	0	17	5	8	>	18	1	2	2	2	2	2	0	3
leanS4 2.0	0	0	0	0	0	0	1	1	2	2	1	0	1	0	1	1	0	0
□KE	8	0	>	>	0	>	6	4	3	3	9	6	4	3	1	>	3	1
LWB 1.0	3	5	11	7	9	>	8	7	8	6	8	6	4	8	4	9	9	12
TA	9	0	>	4	14	0	6	>	9	10	15	>	5	5	>	1	11	0
FaCT 1.2	>	>	4	4	2	>	5	4	8	4	2	1	5	4	>	2	5	3
DLP 3.1	>	>	18	12	>	>	10	>	3	>	15	15	7	>	>	>	>	>

TABLE 3. Results for **S4**

As the table shows, the hardest instances of many of the completely-solved formula classes can be solved in under one second. Allowing for larger values of the parameter would not make these tests effective. In fact, some of the problems are completely or almost-completely solved in the input normalisation phases of the systems. This is shown in Table 5, which gives (for DLP) the maximum total time and time for the search component, as well as the growth with respect to the parameter in search time, backtracks, and number of modal successors visited.[5] As can be seen, most of the tests no longer have exponential growth in the parameter. In fact, many of them

[5] Some of the growth orders are only approximate because of the limited data.

	*SAT 1.2		DLP 3.2		TA 1.4		
Test	Size	Time	Size	Time	Size	SPASS	FLOTTER
branch_p	>	0.21	19	46.06	6	51.95	13.81
branch_n	12	94.49	13	53.63	6	84.21	12.23
d4_p	>	0.06	>	0.05	15	0.64	70.47
d4_n	>	2.87	>	1.12	14	1.14	42.92
dum_p	>	0.04	>	0.02	17	3.32	61.67
dum_n	>	0.12	>	0.02	16	1.75	64.07
grz_p	>	0.04	>	0.04	>	0.35	0.16
grz_n	>	0.01	>	0.05	>	0.16	0.17
lin_p	>	0.01	>	0.03	>	1.03	8.21
lin_n	>	47.80	>	0.13	>	16.07	63.94
path_p	>	0.72	>	0.32	5	22.85	2.18
path_n	>	0.96	>	0.36	4	58.70	2.14
ph_p	8	48.54	7	10.23	6	42.19	0.97
ph_n	12	0.60	>	2.69	9	45.21	9.92
poly_p	>	1.73	>	0.11	5	2.48	51.00
poly_n	>	2.25	>	0.18	4	1.23	7.86
t4p_p	>	0.29	>	0.06	16	3.91	84.75
t4p_n	>	1.28	>	0.13	9	3.37	84.35

TABLE 4: Timing Results from [15] for *SAT (options `-k1 -e -m6`), DLP and TA for **K**. (Courtesy of E. Giunchiglia, F. Giunchiglia and A. Tacchella.)

Test Name	Result	Total Time	Search Time	Search Growth	Backtrack Growth	Successor Growth
branch_p	19	46.06	45.07	$O(2^n)$	0	$O(2^n)$
branch_n	13	53.63	53.50	$O(2^n)$	0	$O(2^n)$
d4_p	>	0.05	0.02	$O(n)$	0	$O(n)$
d4_n	>	1.12	1.08	$O(n^c)$	$O(n)$	$O(n)$
dum_p	>	0.02	0.01	$O(n)$	1	$O(n)$
dum_n	>	0.02	0.01	$O(n)$	0	$O(n)$
grz_p	>	0.04	0.00	$O(c)$	2	$O(c)$
grz_n	>	0.05	0.02	$O(n)$	$O(n)$	$O(n)$
lin_p	>	0.03	0.00	0	0	0
lin_n	>	0.13	0.05	$O(n)$	0	0
path_p	>	0.32	0.25	$O(n)$	0	$O(n)$
path_n	>	0.36	0.28	$O(n^c)$	0	$O(n)$
ph_p	7	10.23	10.21	$O(c^n)$	$O(c^n)$	1
ph_n	>	2.69	0.53	$O(n^c)$	0	$O(n^c)$
poly_p	>	0.11	0.04	$O(n)$	1	$O(n)$
poly_n	>	0.18	0.11	$O(n)$	0	$O(n)$
t4p_p	>	0.06	0.04	$O(n)$	$O(n)$	$O(n)$
t4p_n	>	0.13	0.10	$O(n)$	$O(n)$	$O(n)$

TABLE 5. Growth for DLP for **K**

have linear growth or even no growth at all.

The tests for **K** that remain hard for both *SAT and DLP are *branch_n* and *ph_p*. The first of these consists of the Halpern and Moses branching formulae [20], which have an exponentially-large counter-model but no disjunction. The time taken to build this counter-model is what makes these formulae difficult, and systems that try to store the entire model at once will find these formulae even more difficult. The second is an instance of the Pigeon-Hole principle [31], which has hard propositional reasoning but essentially no modal reasoning.

4.4 Discussion

The Heuerding and Schwendimann benchmarks were designed to meet many of the criteria that we deem important. Both the formulae and their generator were published, along with the rationale behind the formulae, so the tests can be reproduced and extended. The formula classes have a built-in balance between valid and invalid formulae. The test methodology is not computationally intensive and does provide data that can be easily summarised. The size parameter was designed to allow for a monotonic, exponential increase in the difficulty of the formulae. The formula classes attempt to cover a considerable variety of formula structures, thus providing a measure of significance to the test. The formulae are large, but not excessively so. The formulae are artificial, in that they do not correspond to any particular application, but were supposed to be representative of general formulae.

However, with the advent of heavily-optimised provers, many of the formula classes have become too easy under the initial parameter cut-off. Most of these easy formula classes have become in some sense trivial, with input normalisation of the formulae doing most or all of the work. This "triviality" has two effects. First, increasing the parameter will not significantly increase the difficulty of a formula, at least until the formulae become gigantic. Second, these classes no longer test the comparative performance of the search mechanisms, but instead only show that of a particular normalisation. Thus these formula classes, although historically interesting, are no longer good tests for state-of-the-art modal decision procedures, which all employ sufficient input normalisation to render these formula classes "trivial".

An augmented set of formula classes could be created to try to solve the problems detailed above, but it is difficult to devise by hand formulae that are resistant to the various input normalisations of the heavily-optimised systems, and it may prove impossible to devise formulae that will resist new and as yet unknown optimisations.

5 The 3CNF$_{\Box_m}$ Random Tests

Random formulae can be generated that do not have the problems of the Heuerding and Schwendimann formulae. The first random generation technique used in testing modal decision procedures, the random 3CNF$_{\Box_m}$ test methodology, was proposed by Giunchiglia and Sebastiani [16; 18]. It was conceived as a generalisation of the 3SAT test method which is widely used in propositional satisfiability [29]. The method was subsequently criticised and improved by Hustadt and Schmidt [26; 27], who pointed out some major weaknesses, and proposed some solutions. Finally Giunchiglia et.al. [12; 13] proposed the current version of the method, which embeds Hustadt and Schmidt's suggestions, plus some further improvements. (If not otherwise stated, from now on by "random 3CNF$_{\Box_m}$ formulae" we implicitly mean the formulae generated with the final version of the 3CNF$_{\Box_m}$ random generator algorithm described in [13].)

5.1 Description

In the 3CNF$_{\Box_m}$ test methodology, the performance of a system is evaluated on sets of randomly generated 3CNF$_{\Box_m}$ formulae. A CNF$_{\Box_m}$ formula is a conjunction of CNF$_{\Box_m}$ clauses, where each clause is a disjunction of either propositional or modal

literals. A literal is either an atom or its negation. Modal atoms are formulae of the form $\Box_i C$, where C is a CNF$_{\Box_m}$ clause. For normal modal logics there is no loss in the restriction to CNF$_{\Box_m}$ formulae, as there is an equivalence between arbitrary normal modal formulae and CNF$_{\Box_m}$ formulae. [6]

A 3CNF$_{\Box_m}$ formula is a CNF$_{\Box_m}$ formula where all clauses have exactly 3 literals. The definition extends trivially to K-CNF$_{\Box_m}$ formulae, for any positive integer K. Again, there is no loss in restricting attention to 3CNF$_{\Box_m}$ formulae, as there is a satisfiability-preserving way of converting any modal formula into 3CNF$_{\Box_m}$.

In the 3CNF$_{\Box_m}$ test methodology, a 3CNF$_{\Box_m}$ formula is randomly generated according to the following parameters:

- the (maximum) modal depth d;
- the number of clauses L;
- the number of propositional variables N;
- the number of distinct box symbols m;
- the probability p of an atom occurring in a clause at depth $< d$ being purely propositional.

Notice that d represents the *maximum* depth of a 3CNF$_{\Box_m}$ formula: e.g., if $p = 1$, then the depth is 0, no matter the value of d.

The random 3CNF$_{\Box_m}$ generator works as follows:

- a 3CNF$_{\Box_m}$ formula of depth d is produced by randomly, and independently,[7] generating L 3CNF$_{\Box_m}$ clauses of depth d, and forming their conjunction;
- a 3CNF$_{\Box_m}$ clause of depth d is produced by randomly generating three distinct, under commutativity of disjunction, 3CNF$_{\Box_m}$ atoms of depth d, negating each of them with probability 0.5, and forming their disjunction;
- a propositional atom is produced by picking randomly an element of $\{A_1, \ldots, A_N\}$;
- a 3CNF$_{\Box_m}$ atom of depth $d > 0$ is produced by generating a random propositional atom with probability p; and with probability $1 - p$, a 3CNF$_{\Box_m}$ atom $\Box_r C$, where $\Box_r$ is picked randomly in $\{\Box_1, \ldots, \Box_m\}$ and C is a randomly generated 3CNF$_{\Box_m}$ clause of depth $d - 1$.

As there are no repetitions inside a clause, the number $D(d, N)$ of possible distinct, under commutativity of disjunction,[8] 3CNF$_{\Box_m}$, with $m = 1$, atoms of depth d is given by the recursive equation

$$\begin{aligned} D(0, N) &= N \\ D(d, N) &= 2^3 \cdot \binom{D(d-1, N)}{3} \quad [+D(0, N) \text{ if } p \neq 0], \end{aligned} \tag{5.1}$$

That is, $D(d, N)$ grows approximately as $(2N)^{(3^d)}$.

A typical problem set is characterised by a fixed N, m, d and p: L is varied in such a way as to empirically cover the "100% satisfiable—100% unsatisfiable" transition.

[6] The conversion works recursively on the depth of the formula, from the leaves to the root, each time applying to sub-formulae the propositional CNF conversion and the transformation $\Box_r \bigwedge_j \bigvee_i \varphi_{ij} \Longrightarrow \bigwedge_j \Box_r \bigvee_i \varphi_{ij}$.

[7] This means that clauses may be repeated in a formula.

[8] We consider two formulae that differ only in the order of disjuncts in embedded disjunctions to be the same formulae, that is $A \vee B \vee C$ is the same as $B \vee C \vee A$.

Then, for each tuple of the five values in a problem set, a certain number (100, 500, 1000, ...) of $3\text{CNF}_{\Box_m}$ formulae are randomly generated, and the resulting formulae are given as input to the procedure under test, with a maximum time bound of, typically, 1000 seconds. The fraction of satisfiabile formulae, median/percentile values of CPU times, and median/percentile values of other parameters, e.g., number of steps, memory, etc., are plotted against the number of clauses L.

To save testing time, if significantly more than 50% of the samples seen so far exceed the time bound for a given value of L—so that the median and the other Qth percentiles for $Q \geq 50$ are very likely to exceed the bound—then the system is not run on the other samples, and the test goes on with the next L value.

5.2 Parameters and Features

In the random $3\text{CNF}_{\Box_m}$ test method, each of the five parameters plays a specific role.

- d represents the maximum depth of the Kripke models for the $3\text{CNF}_{\Box_m}$ formulae, and thus increases exponentially the size of the potential Kripke models [19]. Thus d allows for increasing at will the difficulty of the solution, particularly for the the modal component of reasoning. The size of generated formulae grows on average as $(3 - 3p)^d$.
- N defines the number of propositional variables which are to be assigned within each Kripke state, and thus enlarges exponentially the space of the possible models for φ. (Typically N is very small with respect to 3SAT problems, as each propositional atom has an "implicit multiplicity" equal to the number of states of a potential Kripke model.) Thus N allows for increasing at will the difficulty of the solution, particularly for the propositional component of reasoning. N has no effect on the size of generated formulae.
- L is the number of conjuncts in the formula, and thus it controls the *constrainedness* of the formula (see, e.g., [36; 8]): the bigger L, the more likely unsatisfiable the formula. Increasing L, we pass with continuity from an initial 100% satisfiability fraction to 100% unsatisfiability. Tuning L allows for balancing the satisfiable vs. unsatisfiable ratio. Obviously the size of generated formulae grows as $O(L)$.
- m represents the number of distinct accessibility relations in the potential models. In $\mathbf{K_{(m)}}$ m partitions each branch in the search tree into m independent sub-branches, each restricted to a single $\Box_r$. Therefore, for larger m the search space is more partitioned and the problem should be easier; moreover, as there is no mutual dependency between the modal satisfiability of the distinct sub-branches, for larger m the formulae are less constrained and more likely to be satisfiable. Thus m affects both difficulty and constrainedness. m has no effect on the size of generated formulae.
- p represents the "propositional vs. modal" rate for the atoms. p has been introduced to unbalance the tree-structure of the random $3\text{CNF}_{\Box_m}$ formulae, and allows for distributing the propositional atoms at the different depth levels. Increasing p reduces the number of modal atoms, and thus reduces the difficulty of the solution, particularly for the modal component of reasoning. Increasing p causes a reduction in size of the formula, as the average size of formulae is $O((3 - 3p)^d)$.

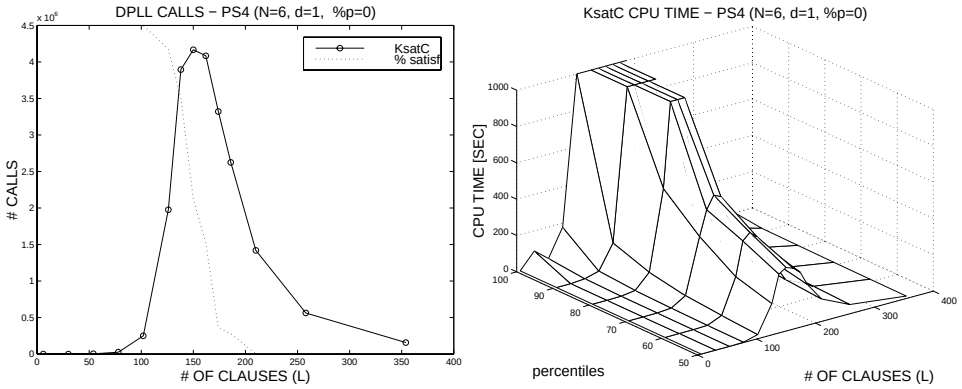


FIG. 1: KSATC on a 3CNF$_{\Box_m}$ test set (d=1, N=6, p=0, m=1, 100 samples/point). (Left): satisfiability fraction and median # of calls; (Right): Qth percentile CPU time.

In a 3CNF$_{\Box_m}$ test session the values of the parameters should be chosen as follows. First, set the values for d, m and p so as to target an area of the input space; then set N to fix the desired level of difficulty; finally, run tests for increasing values of L, so as to cover the whole transition from 100% satisfiability to 0% satisfiability.

Consider the plots in Figure 1, presenting a test for KSATC from [13], with d=1, N=6, p=0, m=1, and 100 samples/point. Figure 1 (left) represents the fraction of satisfiable formulae together with the median number of recursive calls of KSATC, which measures the size of the space searched. Figure 1 (right) represents the Qth percentile CPU time. Both plots exhibit the typical easy-hard-easy pattern centered around the 50% cross-over point—also called "phase transition"—which has been revealed in many NP-complete problems (see, e.g., [29; 36]). This should not be a surprise, as satisfiability in $\mathbf{K}_{(\mathbf{m})}$ and in most modal logics is NP-complete if the modal depth is bounded [19], which is the case of each single plot. Thus, generally speaking, by tuning N and L it is possible to generate very hard 3CNF$_{\Box_m}$ problems with the maximum uncertainty on the results.

It might sound counter-intuitive that, after the 50% cross-over point, the difficulty of the problem decreases with the size of the formula. Giunchiglia and Sebastiani [17] noticed that this is due to the capability of a system to backtrack at any constraint violation: the larger the value of L, the more constraint violations are detected, and the higher the search tree is pruned. As a side-effect, if a plot does not decrease after the 50% cross-over point, then this may reveal a problem of constraint violation detection in the system.

5.3 Problems

As highlighted by Hustadt and Schmidt [26; 27], the formulae generated by the first version of the method suffered from a couple of major drawbacks:

(Propositional) redundancy: As the initial 3CNF$_{\Box_m}$ generator did not check for

repeated propositional variables inside the same clause, the random formulae generated could contain propositional tautologies. When this was the case, the size of formulae could be reduced (in some cases dramatically) by a propositional simplification step;

Trivial (un)satisfiability: For certain values of the generator's parameters, the formulae generated were "trivially (un)satisfiable", that is, they could be solved at the top level by purely propositional inference. (This definition is slightly different from the one in Hustadt and Schmidt, as explained below.)

As a solution, Hustadt and Schmidt [27] proposed three guidelines for generating more challenging problems:

(i) Set to their smallest possible value those parameters that do not significantly influence the difficulty of the $3\mathrm{CNF}_{\Box_m}$ formulae. They included among them the parameters m and d, which they suggested setting to 1.

(ii) To avoid trivially (un)satisfiable formulae, set $p = 0$.

(iii) Modify the random generator so as to prevent repeated propositional variables inside the same clause.

They also improved the data analysis considering not just median values as was done in [16], but a range of Qth percentile values.

Redundancy was a problem due to a flaw in the first version of Giunchiglia and Sebastiani's generator, and it was easily solved. The solution (iii) completely solved propositional redundancy. Giunchiglia et.al. [13] later noticed that an extra component of redundancy was due to the presence inside one clause of repeated *modal* atoms and of permutations of the same modal atom. Thus, they introduced the following variation of (iii):

(iii') Modify the random generator so as to create distinct atoms, under commutativity of disjunction, both propositional and modal, inside the same clause.

Trivial (un)satisfiability is a much more complex problem, and deserves some more discussion.[9]

5.3.1 Trivial Satisfiability

A $3\mathrm{CNF}_{\Box_m}$ formula φ is trivially satisfiable iff it has at least one $\neg\Box$-free satisfying assignment. (Hustadt and Schmidt's definition is in terms of being satisfiable on a Kripke model with one world, which works out the same.) The existence of one $\neg\Box$-only clause is a sufficient condition for avoiding trivial satisfiability, as every assignment then contains at least one $\neg\Box$ literal. Each top-level literal is $\neg\Box$ with probability $(1-p)/2$; each top-level clause is $\neg\Box$-only with probability $(1-p)^3/8$; the probability of having no such clauses is thus $(1-(1-p)^3/8)^L$. As the set of trivially satisfiable formulas is a subset of the set of formulas having no $\neg\Box$-only clause, the probability of being trivially satisfiable is bounded above by a function that converges exponentially to 0 as L goes to infinity. This matches the empirical behaviour revealed in [27], where the fraction of trivially satisfiable formulae decays exponentially with L.

[9] Some discussion about trivial solvability—although to a lower level of detail—can be found in [14].

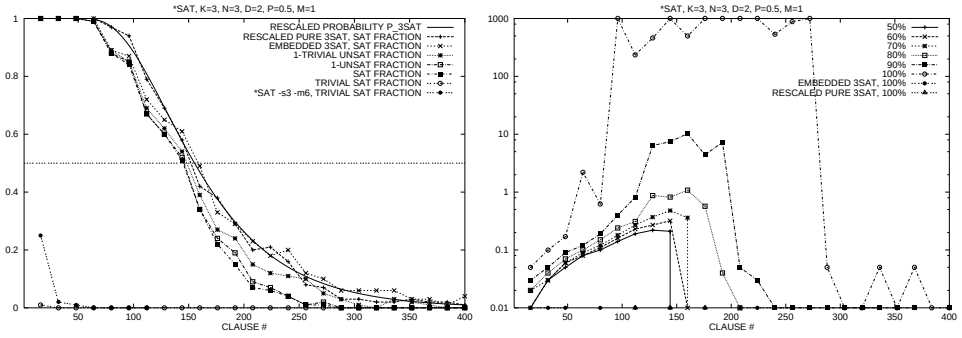


FIG. 2: *SAT on a 3CNF$_{\Box_m}$ test set, (d=2, N=3, p=0.5, m=1, 100 samples/point). (Left): (un)satisfiability rates; (Right): Qth percentile CPU time (seconds), log scale.

Trivial satisfiability is not a big problem for random 3CNF$_{\Box_m}$ testbeds. First, a trivially satisfiable formula is not necessarily trivial to solve. In fact, in the general case, a literal l occurring in a model μ for φ, has the same probability of being positive or negative; thus μ is $\neg\Box$-free with probability 2^{-w}, w denoting the number of modal literals in μ. On average, only a very small percentage of satisfying assignments are $\neg\Box$-free, and unless the procedure is explicitly biased to detect trivial satisfiability there is no reason to assume that the first assignment found by the procedure will be one of them. Biasing the procedure for detecting trivial satisfiability may not be a good strategy in the general case. (For instance, the suggested default settings of the *SAT system are not the best ones to detect trivial satisfiability.) In practice, most current state-of-the-art procedures are not guaranteed to solve trivially satisfiable formulae without any modal reasoning. Second, due to the exponential decrease in the fraction of trivially satisfiable formulae with L, the effects of trivial satisfiability are limited to the extreme left part of the satisfiability plots—where problems are easy and satisfiable anyway—and typically are negligible in the satisfiability transition area—which is the really interesting zone. Moreover, notice that the $(1-(1-p)^3/8)^L$ bound is pessimistic, as typically the fraction decreases much faster. This is due to the fact that for larger values of L, it is harder to get $\neg\Box$-free assignments, even if no $\neg\Box$-only clause occurs in the formula; for instance, the presence of a clause like $(\neg\Box\varphi_1 \vee \neg\Box\varphi_2 \vee \neg A_1)$ forces all assignments which include A_1 to include at least one $\neg\Box$ literal.

Consider for example the plots in Figure 2 (left), which results from running *SAT with its default settings on a 3CNF$_{\Box_m}$ testbed with d=2, N=3, p=0.5, m=1, 100 samples/point. In this test, *SAT found only 3 trivially satisfiable formulae for $L = 16$, and none elsewhere. Even rerunning the testbed with the *SAT option `-s3 -m6` ("while branching, choose always positive values first"), which will find more $\neg\Box$-free assignments, we obtain only 25, 2 and 1 trivially satisfiable formulae for $L = 16$, 24, 32 respectively, and none elsewhere.

5.3.2 Trivial Unsatisfiability

A random $3\text{CNF}_{\Box_m}$ formula φ contains on average Lp^3 clauses that contain three propositional literals (and thus contain no boxes). The larger the value of L, the more likely it is that these clauses will be jointly unsatisfiable—thus making φ trivially unsatisfiable. (The definition of trivial unsatisfiability by Hustadt and Schmidt is precisely this, but our definition also allows unsatisfiability resulting from complementary modal literals in top-level clauses.) This explains the large number of trivially unsatisfiable formulae detected empirically by Hustadt and Schmidt [27] in the center and right of their plots. If $p = 0$, then φ contains no such purely propositional clauses so the problem is eliminated, at least so far as Hustadt and Schmidt's definition is concerned.

When $p > 0$, however, trivial unsatisfiability becomes a serious problem in $3\text{CNF}_{\Box_m}$ testbeds. First, a trivially unsatisfiable formula is typically exceedingly easy to solve for most state-of-the-art procedures because any reasonable $3\text{CNF}_{\Box_m}$ test set for these procedures will have only a very small number of propositional variables. In fact most procedures look for modal successors only after having found a satisfying assignment: if no such assignment exists, the formula is solved without performing any modal reasoning. Second, when p is significantly greater than zero, trivially unsatisfiable formulae may affect all the values beyond the 100% satisfiable range, including the satisfiability transition area. For instance, when $p = 0.5$, it is difficult to generate unsatisfiable $3\text{CNF}_{\Box_m}$ formulae that are not trivially unsatisfiable. A "signature" of the heavy presence of trivial unsatisfiability is what we call a "Half-Dome plot"[10] for median CPU times: the median value grows until L reaches the 50% satisfiability crossover point, then it falls abruptly down to (nearly) zero, and does not increase significantly thereafter. Analogously, the Qth percentile value grows until L reaches the (100-Q)% satisfiability crossover point, and then falls abruptly.

For example, consider Figure 2. In this figure, together with the $3\text{CNF}_{\Box_m}$ plots, we also present the results of running *SAT on the conjunction of the purely propositional clauses of the formulae. We call these propositional conjunctions the embedded 3SAT sub-formulae. Moreover, as a comparison, we also run *SAT on a pure 3SAT testbed with $N = 3$, rescaled horizontally by a $1/p^3$ factor. (E.g., $L = 144$ in the plot means that the pure 3SAT formulas have $144 \cdot 0.5^3 = 18$ clauses.) We call these formulas the rescaled pure 3SAT formulas. As $N = 3$ means that we have only 8 distinct pure 3SAT clauses, it is possible to evaluate the SAT probability exactly, which is given by the following equation:

$$[RESCALED]\ P_{3SAT}(L) = \sum_{k=1}^{8} \binom{8}{k} (-1)^{8-k} (k/8)^{L \cdot p^3}. \tag{5.2}$$

In Figure 2 (left) we plot the fraction of satisfiable $3\text{CNF}_{\Box_m}$ formulae, 1 minus the fraction of unsatisfiable $3\text{CNF}_{\Box_m}$ formulae, and 1 minus the fraction of trivially unsatisfiable $3\text{CNF}_{\Box_m}$ formulae. For the embedded 3SAT formulae and the rescaled pure 3SAT formulae, we plot only the fraction of satisfiable formulae, which in both cases is identical to 1 minus the fraction of unsatisfiable formulae. We also plotted the (rescaled) 3SAT probability of Equation 5.2. Needless to say, the two latter plots match apart for some noise. Finally, we plotted the fraction of formulae found to be

[10] The name refers to the shape of a famous mountain in Yosemite, California.

trivially satisfiable by *SAT and by *SAT `-s3 -m6`, as discussed above. (Considering the well-known satisfiability transition results [29; 5], for the rescaled pure 3SAT plot one might expect an average of 50% unsatisfiable formulae for $L \approx 4.28 \cdot N/p^3 = 102.72$; however, $N = 3$ is too small for applying these results, whilst it is possible to provide an exact calculation as in Equation 5.2.) Figure 2 (right) presents run times for the tests.

Several conclusions are evident from this data. Firstly, the fraction of satisfiable 3CNF$_{\Box_m}$ formulae nearly coincides with 1 minus the fraction of unsatisfiable 3CNF$_{\Box_m}$ formulae that is, very few tests exceeded the bound. Secondly, the fraction of unsatisfiable 3CNF$_{\Box_m}$ formulae and the fraction of trivially unsatisfiable 3CNF$_{\Box_m}$ formulae are very near, that is, most unsatisfiable formulae are also trivially unsatisfiable. (In our testing experience, unsatisfiable 3CNF$_{\Box_m}$ formulae that are not trivially unsatisfiable are rare in testbeds with $p = 0.5$.) Thirdly, the fraction of unsatisfiable embedded 3SAT formulae is very close to the fraction of trivially unsatisfiable 3CNF$_{\Box_m}$ formulae, that is, most trivially unsatisfiable formulae are such because the embedded 3SAT component is unsatisfiable. Fourthly, the fraction of unsatisfiable embedded 3SAT formulae is very close to the fraction of unsatisfiable rescaled pure 3SAT formulae.

In Figure 2 (right), the plots for the embedded 3SAT and for the rescaled pure 3SAT testbeds cannot be distinguished from the X axis—that is, CPU times are always smaller than 0.01 seconds. The median CPU time plot for the 3CNF$_{\Box_m}$ testbed grows until L reaches the 50% satisfiability crossover point, where it falls down abruptly; the remainder of the plot cannot be distinguished from the X axis. The Qth percentile values behave analogously with respect to the (100-Q)% satisfiability crossover point.

This behaviour can be explained as follows. If we plot the fraction of satisfiable formulae for the embedded 3SAT formulae, we obtain a satisfiability transition similar to the one of the rescaled pure 3SAT, with a 50%-satisfiable crossover point for $L \approx 150$. The slight differences between the two plots are due to the fact that the length of the embedded 3SAT formulae is not fixed with L, as $p^3 \cdot L$ is only an average value. As N is generally rather small, the embedded 3SAT formula is mostly unsatisfiable with a very low value of L, making the entire formula trivially unsatisfiable. Thus the embedded 3SAT transition dominates the whole satisfiability plot. The first effect is that the transition is expected to nearly coincide with the embedded 3SAT transition. The second effect is that nearly all unsatisfiable formulae are trivially solvable. This means that the CPU times are entirely dominated by the values required to solve satisfiable formulae, which typically grow with L. Immediately after the (100-Q)% satisfiability crossover point, the easiest Q% of samples are nearly all trivially unsatisfiable, so that the Qth percentile value falls down abruptly to a negligible value.

The small difference between the fraction of the embedded 3SAT formulae that are unsatisfiable and the fraction of the 3CNF$_{\Box_m}$ formulae that are trivially unsatisfiable can be explained as follows. When the embedded 3SAT formula is "nearly unsatisfiable", that is, it has only one or very few models, some other clauses may contribute to cause trivial unsatisfiability. For instance, consider $\varphi = \varphi_1 \wedge (C_1 \vee \Box\psi) \wedge (C_2 \vee \neg\Box\psi)$, where C_1 and C_2 are propositional sub-clauses, and the embedded 3SAT formula, φ^*, is "nearly unsatisfiable". If the few models of φ^* each violate both C_1 and C_2, then all assignments must propositionally satisfy $\varphi^* \wedge (\Box\psi) \wedge (\neg\Box\psi)$. If $\Box\psi$ is treated as an atom (which is the case in most optimised systems), then φ is trivially unsatisfiable, even though φ^* is satisfiable. With $p = 0.5$, an average of 3/8 clauses have exactly

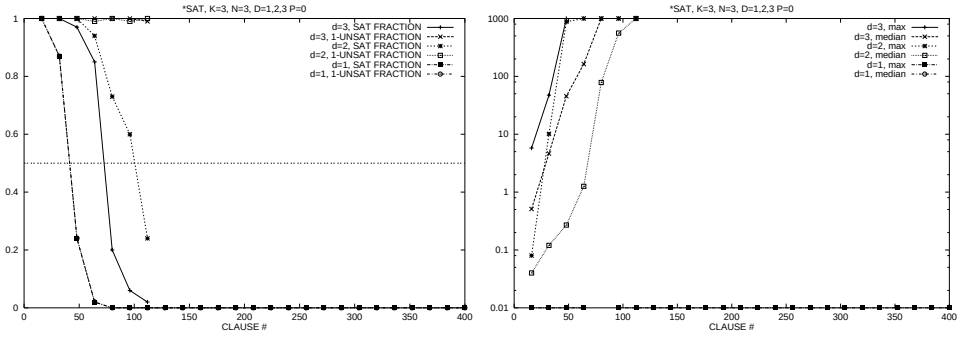


FIG. 3: A 3CNF$_{\Box_m}$ random test set (d=1,2,3, N=3, p=0, m=1, 100 samples/point). (Left): (un)satisfiability rates; (Right): median and max CPU times (seconds) for *SAT, log scale.

one modal literal. With $N = 3$ and $d = 2$, the probability that two such clauses have mutually contradictory modal literals $\Box\psi$ and $\neg\Box\psi$ is not negligible. As before, with $p = 0$ no "nearly unsatisfiable" embedded 3SAT formula occurs.

5.3.3 The $p = 0$, $d = 1$, $m = 1$ Solution

Unfortunately the guidelines indicated by Hustadt and Schmidt [27], and described above, are not a panacea, as they introduce new problems.

Consider the $d = 1$ guideline. First, in $\mathbf{K_{(m)}}$ the 3CNF$_{\Box_m}$ class represents only the class of formulae of depth d, as there is no way to reduce the depth of formulae. Therefore, if $d = 1$ the input subspace sampled is not very representative. Moreover, as shown in [19], a formula φ which is satisfiable in $\mathbf{K_{(m)}}$ (and also in $\mathbf{KT_{(m)}}$, $\mathbf{K}45_{\mathbf{(m)}}$, $\mathbf{KD}45_{\mathbf{(m)}}$ and $\mathbf{S5_{(m)}}$) has a tree-like Kripke model whose number of states is smaller than $|\varphi|^{depth(\varphi)}$, where $|\varphi|$ and $depth(\varphi)$ are respectively the size and the modal depth of φ. As a consequence, satisfiable 3CNF$_{\Box_m}$ formulae with $d = 1$ have very small models, so that they are not very challenging from the viewpoint of pure modal reasoning, regardless of the values chosen for the other parameters. More generally, when bounding the modal depth, the satisfiability problems for the logics above decays from PSPACE-complete to NP-complete [19].

Consider the $p = 0$ guideline. If $p = 0$, then the random 3CNF$_{\Box_m}$ formulae are complete ternary trees where propositional atoms occur only at the maximum depth level. Such formulae can hardly be considered as a representative sample of the modal input space. Moreover, they are even less representative if used as a testbed for most modal logics different from $\mathbf{K}$. In fact, in most modal logics, restricting the occurrence of propositional variables to the maximum depth level hinders a relevant source of reasoning due to the interaction between variables occurring at different depth levels. For instance, the assignment $\{A_1, \Box_1\varphi\}$ is satisfiable in $\mathbf{K_{(m)}}$ but may be not in $\mathbf{KT_{(m)}}$ if A_1 occurs in φ (e.g., if $\varphi = \neg A_1$).

Finally, consider the $d = 1$ and $p = 0$ guidelines together. In [27] the $d = 1$ statement derived from the results of an experiment with $N = 3$, $p = 0.5$, $m = 1$,

$d = 2, 3, 4, 5$ where the complexity did not seem to increase significantly with d, and the growth was smaller than the increase in size. Unfortunately, this experiment was strongly influenced by the $p = 0.5$ choice. With $p = 0$, it turns out that the overall difficulty grows dramatically with d. Consider the plots in Figure 3, which have been obtained by running *SAT on three test sets, with $d = 1$, 2 and 3, $N = 3$, $p = 0$, m =1, and 100 samples/point. Figure 3 (left) shows both the fraction of formulae found to be satisfiable and 1 minus the fraction of formulae found to be unsatisfiable. Figure 3 (right) shows the median and max CPU times. For $d = 1$, all formulae are either found to be satisfiabile or found to be unsatisfiable—that is, no sample exceeded the timeout—and the satisfiability fraction decreases very fast, reaching 100% unsatisfiability for $L < 100$; the median and max CPU times are so small that cannot be distinguished from the X axis. This should not be a surprise: with $N = 3$, $p = 0$ and $d = 1$, there are only 8 distinct modal atoms, so that the test bed is only a little harder than a 3SAT testbed with $N = 8 + 3$ variables. For $d = 2, 3$ things change dramatically. Both median and maximum CPU times rapidly reach the timeout. As a consequence, the fraction of formulae found to be satisfiable plus the fraction of formulae found to be unsatisfiable add to much less than 1, as most problems exceed the timeout. The real satisfiability fraction is somewhere between them. For instance, with $L = 80$ more than 50% of the $d = 3$ samples have exceeded the timeout, while for $L = 120$ more than 50% of the $d = 2$ samples have exceeded the timeout. Running *SAT for $d = 2$ and $L = 400$, no sample was solved within the timeout. As a consequence, for $p = 0$ and $d > 1$, the test sets are mostly out of the reach of *SAT, and no information about the satisfiability transition can be provided. The situation is no better with other current decision procedures such as DLP. To our knowledge, no system so far has been able to fully plot the satisfiability transition for $d = 2$, $N = 3$, $p = 0$, $m = 1$, as in the satisfiability transition area most solution times exceed the timeout.

We believe that such behaviour should be expected. As for satisfiability fraction, the number of possible distinct atoms $N(d, N)$ in equation (5.1) grows exponentially with d, decreasing the probability of conflicts between modal literals. As a consequence, with L fixed, the fraction of unsatisfiable formulae decreases very rapidly with d, causing a relative increase in the number of clauses L necessary to reach the 50% cross-over point, that is, a relative shift to the right of the transition area. As for difficulty, the size of the Kripke models for φ is up to $|\varphi|^{depth(\varphi)}$ [19]. Thus, if L is fixed and d grows, it is reasonable to expect that the effort required to search for such exponentially-big models grows at least exponentially with d. Notice that, if φ is a random 3CNF$_{\Box_m}$ formula with $p = 0$, then $|\varphi|$ also grows as $O(3^d)$.

This prompts the question as to why the same behaviour is not observed with $p = 0.5$. As for satisfiability, the effects of increasing d are counteracted by the embedded 3SAT component, which forces trivial unsatisfiability with very low values of L, preventing the shifting of the satisfiability transition described above. As for difficulty, first, $|\varphi|$ is much smaller, as only about a $(1 - p)^d$ fraction of the branches of the formula tree of φ actually reach depth d. Moreover, the high percentage of propositional literals reduces dramatically the number of assignments found, as it gets much harder for most candidate assignments to avoid containing propositional contradictions like A_j, $\neg A_j$. Finally, the modal literals are only a subset of each assignment found, so that the number of states to be explored for every assignment

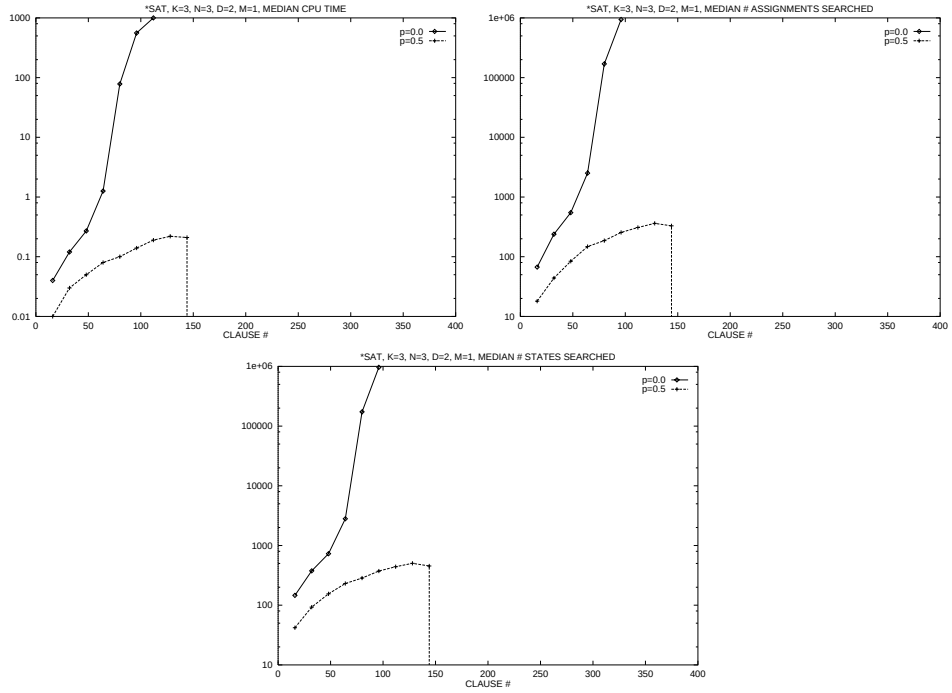


FIG. 4: Two $3\text{CNF}_{\Box_m}$ random test set, $d = 2$, $N = 3$, $p = 0.0$ and 0.5, $m = 1$, 100 samples/point. (Left Top): median CPU time; (Right Top): median # of assignments found; (Bottom): median # of states explored.

is reduced. Figure 4 shows both the median CPU times, the median number of assignments found and the median number of states explored by *SAT for $d = 2$, $N = 3$, $m = 1$, with both $p = 0.0$ and 0.5. It can be seen that all three values are drastically reduced by setting $p = 0.5$.

5.4 Discussion

Even with the above problems, the random $3\text{CNF}_{\Box_m}$ test methodology produces a good empirical test for many purposes. The generators are available and their "randomness" can be controlled by setting the seed of their random number generator to reproduce test sets if the actual formulae are not available. The rationale behind the methodology has been extensively discussed. The formulae generated, although large, are not too large, and many very difficult (relative to size) formulae are generated if appropriate parameter values are chosen. The test methodology produces a balance between satisfiable and unsatisfiable formulae. With the recent improvements, the problems of redundancy and triviality are much reduced.

The test generator is highly parameterised, perhaps too highly, but by concentrating on a subsection of the test space interesting tests can be generated. The biggest problem with the test methodology is that the maximum modal depth of formulae is

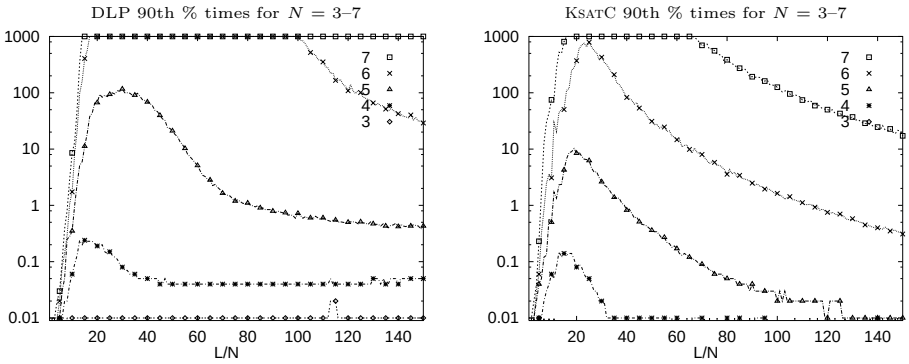


FIG. 5. Results for $m = 1$, $d = 1$, and $p = 0.0$

fixed, thus reducing the inherent difficulty of the problem from PSPACE-complete to NP-complete. However, even at modal depths 1 and 2, difficult tests can easily be devised.

One disadvantage with random tests is that they take much longer to perform (for interesting hard problems) because a large number of formulae have to be generated and tested at each data point for the results to be reliable. However, probably the biggest drawback with the random 3CNF$_{\Box_m}$ test methodology is that, because of their low modal depths, the formulae generated are very artificial. This is not really a problem with the test methodology *per se*, but is instead due to the combination of the test methodology and the capabilities of current decision procedures.

One benefit of the random 3CNF$_{\Box_m}$ test methodology is that it can show the changing relative behaviour of several systems as the various parameters change. For example, several qualitative differences between DLP and KSATC can be discerned from the tests shown in Figures 5, 6, and 7 which give 90th percentile results for several tests.

These results illustrate a number of differences between the two decision procedures, which can be traced back to characteristics of the system. For example, KSATC uses an underlying satisfiability engine with very efficient data structures whereas DLP does not have as highly optimised data structures. This difference in data structures shows up in different run times for larger formulae (larger values of L/N) with non-zero p (Figures 6 and 7) where formulae are mostly trivially unsatisfiable, but where DLP takes some amount of time just to traverse its data structures.

KSATC uses an aggressive look-ahead technique that investigates modal successors very early on in the search space. This technique is good when the problems are over-constrained, resulting in better performance in particular for $d = 1$ and $p = 0$ (Figure 5), and also in narrower peaks for $d = 1$ and $p = 0.5$ (Figure 6). However, when there are significant numbers of modal successors that are satisfiable, this early investigation, and the necessary reinvestigation when more information is known, becomes a serious liability, as shown for $d = 2$ and $p = 0.5$ (Figure 7).

It is this sort of comparative analysis that is most useful to the understanding of how various algorithms behave and how they can be improved.

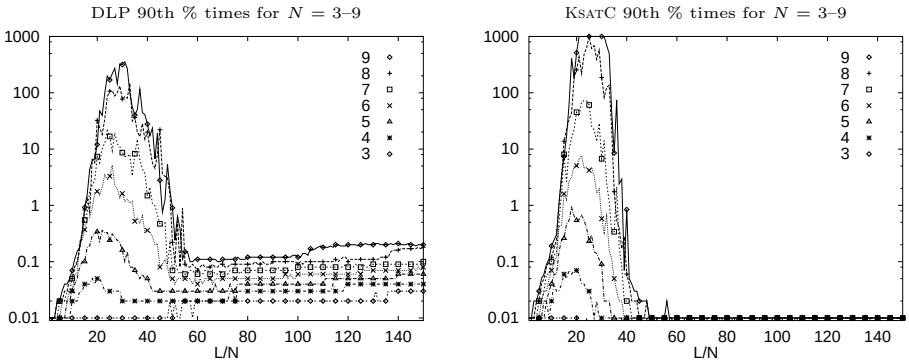


FIG. 6. Results for $m = 1$, $d = 1$, and $p = 0.5$

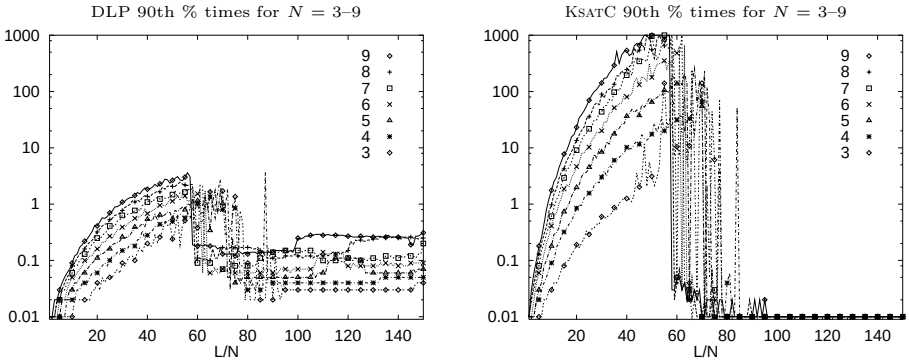


FIG. 7. Results for $m = 1$, $d = 2$ and $p = 0.5$.

6 New Random Empirical Testing

Since 1998 some new forms of random testing—both variants of the $3\text{CNF}_{\Box_m}$ method and completely new ones—have been investigated and/or proposed. In the following section we briefly outline and review the most significant of these.

6.1 Using Modalised Atoms

Recently Massacci [28] proposed a "**K**-modalised" variant of the $3\text{CNF}_{\Box_m}$ method borrowing an idea from [19]: within each $3\text{CNF}_{\Box_m}$ formula φ, substitute each occurrence of each propositional variable A_i with the corresponding modal expression $\neg\Box(A_0 \vee \Box^i \neg A_0)$. (A similar encoding was proposed for **S4**.) The encoding preserves satisfiability in **K**, and the resulting formula φ' has only one propositional variable A_0 and depth $d + N + 1$. Moreover, φ' is relatively bigger than φ, as the global number of propositional literals is doubled and, for each propositional atom, one "$\vee$" and an average of $N/2 + 1$ "$\Box$"s are added. Thus, we would expect **K**-modalised $3\text{CNF}_{\Box_m}$ formulae to be much harder than their corresponding $3\text{CNF}_{\Box_m}$ ones, especially for low d's and high N's.

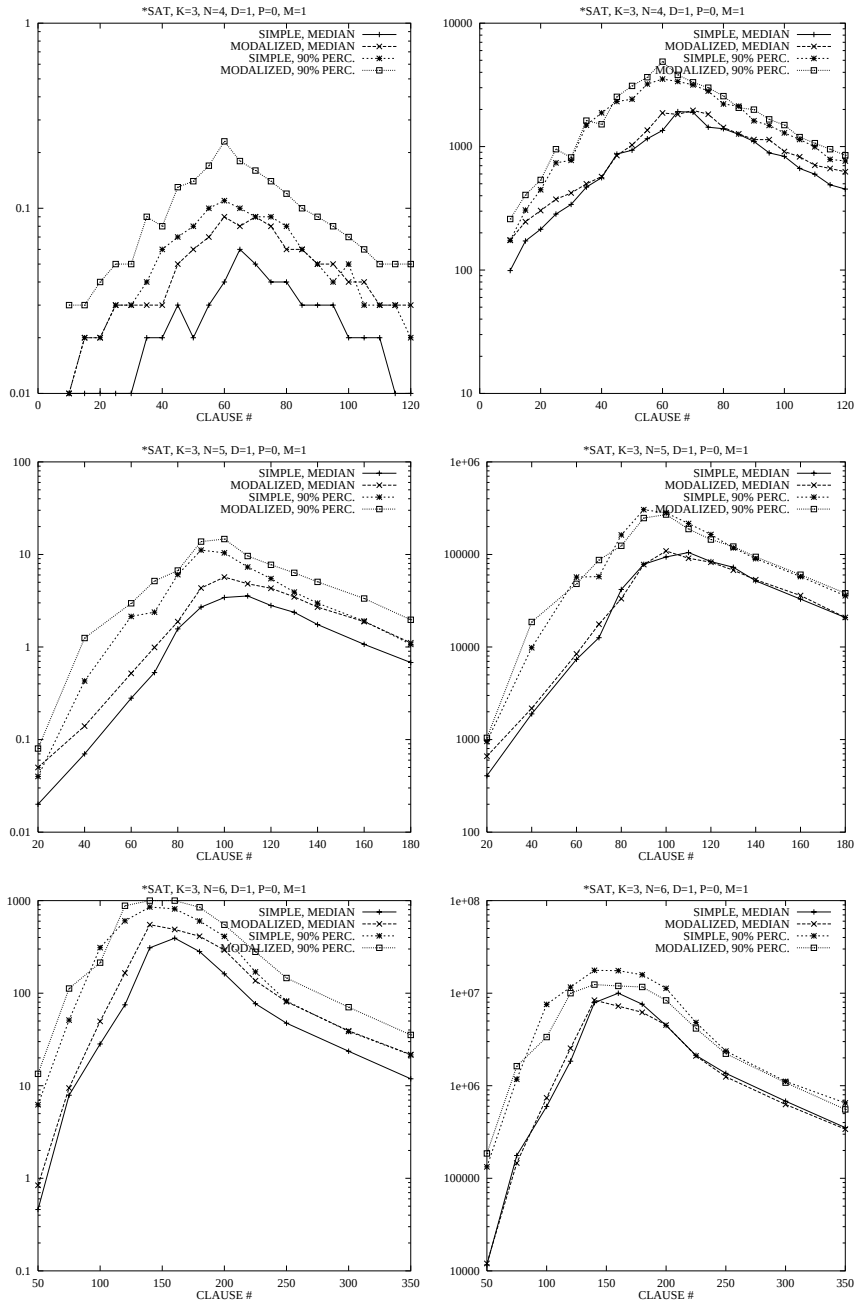


FIG. 8: *SAT on 3CNF$_{\Box_m}$ and **K**-modalised 3CNF$_{\Box_m}$ test sets, (N=4,5 and 6, d=1, p=0, m=1, 100 samples / point). (Left): median and 90% percentile CPU time (seconds). (Right): median and 90% size of the space searched.

Figure 8 shows the results of running *SAT on the testbeds in [27; 13]—i.e., with d=1, p=0, m=1, N=4,5,6 and 100 samples / point—using both $3\text{CNF}_{\Box_m}$ formulae and their **K**-modalised counterparts.The top row represents median and 90% percentile CPU time; the bottom row represents median and 90% percentile size of the space searched, that is, the number of single truth-value assignments performed by *SAT. All curves present the usual easy-hard-easy pattern. From the first row, we notice that the **K**-modalised samples in general require a longer CPU time to solve. Nevertheless, the gap never exceeds a 2-3 factor, which is well justified by the increase in size of the input formulae. (Similarly, the CPU time gap between the **K**-modalised and non-modalised $3\text{CNF}_{\Box_m}$ tests presented in [28] never exceeds a 2-3 factor.) Moreover, from the second row, we notice that there is no significant difference in the size of the space effectively explored.

These results may be explained as follows. As far as basic propositional reasoning is concerned, modalisation introduces no difference, as *SAT considers boxed formulae as propositional atoms. In the simple $3\text{CNF}_{\Box_m}$ case, *SAT takes one shot to determine the satisfiability of a purely propositional assignment like

$$\{A_{i_1}, ..., A_{i_n}, \neg A_{j_1}, ..., \neg A_{j_m}\}. \tag{6.1}$$

In the **K**-modalised case, the assignment corresponding to (6.1) is

$$\{\neg\Box(A_0 \vee \Box^{i_1}\neg A_0), ..., \neg\Box(A_0 \vee \Box^{i_n}\neg A_0), \Box(A_0 \vee \Box^{j_1}\neg A_0), ..., \Box(A_0 \vee \Box^{j_m}\neg A_0)\}. \tag{6.2}$$

Although (6.2) is satisfiable in **K** [19], checking its satisfiability requires determining the satisfiability of the sub-formulae

$$\neg A_0 \wedge \neg\Box^i\neg A_0 \wedge (A_0 \vee \Box^{j_1}\neg A_0) \wedge ... \wedge (A_0 \vee \Box^{j_m}\neg A_0), \tag{6.3}$$

for every negated boxed atom $\neg\Box(A_0 \vee \Box^i\neg A_0)$ in (6.2). However, this step is deterministic. First, all the A_0 disjuncts are wiped off by unit-propagating the first $\neg A_0$:

$$\neg\Box^i\neg A_0 \wedge \Box^{j_1}\neg A_0 \wedge ... \wedge \Box^{j_m}\neg A_0. \tag{6.4}$$

As (6.4) and all its modal successors contain only one negated box, *SAT solves (6.4) deterministically by generating a linear concatenation of i states, without performing any search within each of them. An analogous behaviour is to be expected, e.g., from DLP and FaCT.

As far as we can see from the experiments, despite the relative increase in depth and size of the formulae, **K**-modalisation does not seem to produce testbeds which are significantly more challenging than standard $3\text{CNF}_{\Box_m}$ ones. For instance, in the tests in Figure 8, modalisation simply introduces an overhead due to an extra amount of deterministic steps, without increasing significantly the size of the search space.

6.2 Re-interpreting p: the New_3CNF$_{\Box_m}$ Test Method

To overcome the problems of the $3\text{CNF}_{\Box_m}$ generator due to the embedded SAT component φ^*, we propose a new variant of the random generator of [13], called New_$3\text{CNF}_{\Box_m}$. The difference relies on a different interpretation of the p parameter. In the $3\text{CNF}_{\Box_m}$ generator, p is interpreted as "*the probability of an atom being propositional*". In the New_$3\text{CNF}_{\Box_m}$ generator, p is interpreted as "*the proportion of propositional atoms in a clause*", in the sense that

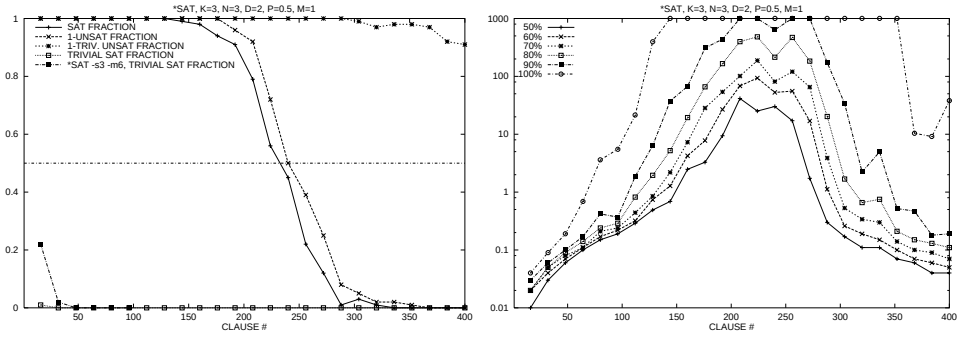


FIG. 9: *SAT on a New_3CNF$_{\Box_m}$ test set, (d=2, N=3, p=0.5, m=1, 100 samples / point). (Left): (un)satisfiability fractions; (Right): Qth percentile CPU time (seconds), log scale.

- if $p = k/3,\ k \in \{0, 1, 2, 3\}$, then the proportion is interpreted in the obvious way, that is, "*exactly k propositional literals and* $3 - k$ *modal literals*".
- otherwise, the residual part is interpreted as a probability, that is, "*exactly* $\lfloor 3p \rfloor$ *propositional literals,* $3 - \lceil 3p \rceil$ *modal literals, and the last literal is propositional with probability* $3p - \lfloor 3p \rfloor$", where $\lfloor x \rfloor =_{def} max\{n \in \mathcal{N} | n \leq x\}$ and $\lceil x \rceil =_{def} min\{n \in \mathcal{N} | n \geq x\}$.

The first case is a sub-case of the second: if $p = k/3$, then $\lfloor 3p \rfloor = \lceil 3p \rceil = 3p = k$. The definition trivially extends to K-CNF$_{\Box_m}$ formulae by substituting 3 with K. A random New_3CNF$_{\Box_m}$ clause is thus generated in the following way (and then sorted):

1. generate randomly $\lfloor 3p \rfloor$ distinct propositional literals;
2. generate randomly $3 - \lceil 3p \rceil$ distinct New_3CNF$_{\Box_m}$ literals;
3. flip a coin: with probability $3p - \lfloor 3p \rfloor$, generate randomly a fresh propositional literal; otherwise, generate randomly a fresh New_3CNF$_{\Box_m}$ literal ("fresh" here means "not already present in the clause").

For instance, if $p = 1/3$, then the clause contains 1 propositional and 2 modal literals; if $p = 0.5$, then it contains 1 propositional and 1 modal literal, and the other is propositional with probability 0.5; if $p = 0.6$, then it contains 1 propositional and 1 modal literals, and the other is propositional with probability 0.8, as $3 \cdot 0.6 - \lfloor 3 \cdot 0.6 \rfloor = 1.8 - 1 = 0.8$.

As with the 3CNF$_{\Box_m}$ case, a New_3CNF$_{\Box_m}$ clause contains an average of $3p$ propositional literals. However, if $p < 2/3$, then no purely propositional clause can be generated. This prevents a random New_3CNF$_{\Box_m}$ formula φ from containing any embedded 3SAT sub-formulae φ^*, and thus eliminates the main source of trivial unsatisfiability while preserving the benefits of setting $p > 0$.

Figure 9 shows plotted the results of running *SAT on a random New_3CNF$_{\Box_m}$ test set, with $d = 2$, $N = 3$, $p = 0.5$, $m = 1$ and 100 samples / point. Figure 9 (left) shows the fraction of formulae found to be satisfiable, 1 minus the fraction of formulae found to be unsatisfiable, 1 minus the fraction of formulae found to

be trivially unsatisfiable, the fraction of formulae found to be trivially satisfiable by *SAT and by *SAT `-s3 -m6`. First, the satisfiable and unsatisfiable fractions add to close to 1 but not exactly to 1, that is, a few tests exceeded the bound. Secondly, there are very few trivially unsatisfiable formulae, but there are some. The trivial unsatisfiability in these formulae is caused by complementary modal atoms in top-level clauses. Thirdly, the fraction of unsatisfiable formulae is much greater than the fraction of trivially unsatisfiable formulae, that is, very few unsatisfiable formulae are also trivially unsatisfiable, and nearly all of these are located in the 100% unsatisfiable zone. With its default settings, *SAT found only 1 trivially satisfiable formula for $L = 16$, and none elsewhere. As before, rerunning the testbed with the *SAT option `-s3 -m6` we obtained 22 and 2 trivially satisfiable formulae for $L = 16$ and 24 respectively, and none elsewhere. In Figure 9 (right), the median CPU time and the other Qth percentile plots reveal a typical easy-hard-easy pattern centred in the satisfiability transition area, growing up to the 50% cross-over point, and then decreasing gently.

Now compare the plots with those of the analogous $3\text{CNF}_{\Box_m}$ test set in Figure 2 (the parameters' values and the system tested are the same). Firstly, the satisfiability transition here is relatively shifted to the right. Secondly, the New_$3\text{CNF}_{\Box_m}$ CPU time plots are slightly harder than the $3\text{CNF}_{\Box_m}$ plots in the satisfiable area, becoming dramatically harder as the fraction of trivially unsatisfiable $3\text{CNF}_{\Box_m}$ formulae increases. In fact, the effect is due to the Qth percentile values for the $3\text{CNF}_{\Box_m}$ test set becoming dramatically lowered by the increasing percentage of trivially unsatisfiable formulae, the solution times for which are negligible. This does not happen with the New_$3\text{CNF}_{\Box_m}$ test set, where the percentage of trivially unsatisfiable formulae is negligible and hard unsatisfiable formulae are generated. Finally, the New_$3\text{CNF}_{\Box_m}$ CPU time plots describe easy-hard-easy patterns which decrease gently with L, instead of falling down abruptly. The New_$3\text{CNF}_{\Box_m}$ plots are no longer dominated by the trivially unsatisfiable formulae, and thus they are free to follow their "natural" easy-hard-easy pattern induced by the increase in the constrainedness.

On the whole, the New_$3\text{CNF}_{\Box_m}$ test method with $p \leq 2/3$ solves the problems related to trivial unsatisfiability in the $3\text{CNF}_{\Box_m}$ method. This is done without imposing the $p = 0$ and $d = 1$ restrictions, which introduce new problems of their own. Work is still ongoing to plot full diagrams for $d > 2$. Generating full New_$3\text{CNF}_{\Box_m}$ plots for bigger values of d and N may be a challenge for both state-of-the art and future systems.

6.3 Random QBF Tests

A common criticism of $3\text{CNF}_{\Box_m}$ testbeds comes from the consideration that for many modal logics the class of formulae with bounded depth is in NP [19], so that $3\text{CNF}_{\Box_m}$ testbeds with bounded d have only NP complexity [28]. To overcome this problem, Massacci proposes a completely new kind of random empirical benchmark, which was used in the TANCS'99 system performance comparison. In this benchmark, random QBF formulae are generated according to the method described by Cadoli et.al. [4], and then converted into modal logic by using a variant of the conversion by Halpern and Moses [20]. The converted modal formulae are satisfiable iff the QBF formulae are true.

Random QBF formulae are generated with alternation depth D and at most V variables at each alternation. The matrix is a random propositional CNF formula with C clauses of length K, with some constraints on the number of universally and existentially quantified variables within each clause. (This avoids the problem of trivial unsolvability for random QBF formulae highlighted by Gent and Walsh [10].) For instance, a random QBF formula with $D = 3$, $V = 2$ looks like:

$$\forall v_{32} v_{31}.\exists v_{22} v_{21}.\forall v_{12} v_{11}.\exists v_{02} v_{01}.\psi[v_{32}, ..., v_{01}]. \tag{6.5}$$

In this section, ψ is a random QBF formula with parameters C, V and D, while U and E denote the total number of universally and existentially quantified variables respectively. Clearly, both U and E are $O(D \cdot V)$. Moreover, φ is the modal formula resulting from Halpern and Moses' **K** conversion, so both the depth and the the number of propositional variables of φ are also $O(D \cdot V)$.

A from-scratch empirical evaluation of the random QBF test method would require an effort exceeding the proposed scope (and length) of this paper. Thus we will restrict our analysis to some basic considerations.

As with $3\text{CNF}_{\Box_m}$, the results of each QBF-based testbed are easy to reproduce as the generator's code and all the parameters' values are publicly available. By increasing V and D the difficulty of the generated problems scales up, while C allows for tuning the sat-versus-unsat rate of the formula. Random QBF plots, with fixed D and V, also present easy-hard-easy patterns centred in the solvability transition areas [4; 10]. Moreover, with respect to $3\text{CNF}_{\Box_m}$ formulae, random QBF formulae allow for generating 50%-solvable formulae with higher modal depth that are still within the reach of current state-of-the-art deciders.

From a purely theoretical viewpoint, it is claimed that, unlike $\text{CNF}_{\Box_m}$ formulas, modal-encoded QBF formulas can capture the problems in Σ_D^P, as QBF formulas with bounded D and unbounded V are in Σ_D^P, while $\text{CNF}_{\Box_m}$ formulas with bounded d and unbounded N are "stuck at NP" [28]. We notice that this statement is rather misleading. Massacci treats the alternation depth D and the number of variables within each alternation V as the "QBF-equivalent" of the modal depth d and the number of propositional variables N respectively – they are the same parameters in the random generator. We remark that the "QBF-analogous" role of modal depth is played instead by the total number of universally quantified variables $U \approx V \cdot D/2$. In fact, similarly to modal $\mathbf{K_{(m)}}$ with bounded depth, the class of random QBF formulae with bounded U is in NP, as it is possible to "guess" a tree-like witness with $O(U \cdot 2^U)$ nodes.[11] Moreover, as in the case of $3\text{CNF}_{\Box_m}$ formulae with bounded d and N, if D and V are bounded—which is the case of every finite-size test set—then the random QBF problems are not only in NP, but even in P.

Another problem with modal-encoded QBF formulae is that they are rather artificial, as their potential Kripke structures are restricted to those having the very regular structure imposed by the QBF and/or binary search trees. As far as representativity is concerned, modal-encoded QBF formulae have a very peculiar modal structure, so that they can hardly be considered as a representative sample of the input space.

Finally, a serious problem with random modal-encoded QBF formulas is size. Initial versions of the translation method produced test sets in the 1GB range. The problem

[11] More precisely $O(E \cdot 2^U)$ nodes, but E is $O(U)$ in the class of formulae considered.

with such very large formulae is that they may be only stressing the data-storage and retrieval portion of the provers; e.g., running DLP on these formulae resulted in a 1000s timeout without any significant search. Even the current versions produce very large modal formulae, mostly to constrain the Kripke structures.

7 Discussion

The current situation in empirical testing of modal decision procedures is not completely satisfactory. There are a number of available test sets and methodologies that can be used to examine modal decision procedures, each of which has certain benefits and certain flaws.

The Heuerding and Schwendimann test suite provides a number of interesting inputs. The input formula classes are different from each other, but there is only a small number of classes and they do not cover all kinds of input formula. The input formulae are parameterised, potentially providing a good range of difficulty. However, current systems incorporate pre-processing steps that reduce many of the test formula classes to trivial formulae even before search starts, dramatically reducing the difficulty of the class.

The standard $3\text{CNF}_{\Box_m}$ random test methodology provides a means for easily generating very hard problems. It has some problems with trivial (un)satisfiability, but these problems are not severe. Disguising the $3\text{CNF}_{\Box_m}$ formulae by modalising the propositional atoms does not make the tests appreciably better. Our new random test methodology may help alleviate these problems, but more testing is required.

The new QBF random testing, when compared with $\text{New\_3CNF}_{\Box_m}$ formulae, allows for generating 50%-solvable formulae with higher modal depth which are still within the reach of current state-of-the-art deciders. On the other hand, modal-encoded QBF formulae are rather artificial, as their potential models are restricted to those having a very regular structure.

Unfortunately, current systems can only handle very small values for some of the parameters at interesting points in the parameter space of the $3\text{CNF}_{\Box_m}$ random test methodology. This makes it very hard to investigate the behaviour of systems over an interesting range of the parameter. Further, there are four active parameters, which makes it hard to cover large sections of the input space.

However, even with the above-mentioned flaws, current empirical test methodologies provide evidence of great strides forward in the performance of modal decision procedures. The current fastest systems, including DLP and *SAT, can quickly solve problems that were impossible to solve just a couple of years ago. The Heuerding and Schwendimann test suite shows the importance of pre-processing to remove as many redundancies as possible. The standard $3\text{CNF}_{\Box_m}$ test suites show the effect, both positive and negative, of certain strategies built into these systems.

It is interesting to compare the situation in empirical testing of modal decision procedures with that for propositional satisfiability [9; 6; 33], where a random-3CNF methodology is also used. With propositional formulae, the methodology can be tuned to provide problems of appropriate difficulty, and captures the hardest formulae of a particular size. There are only two parameters, which allows for easy coverage of large sections of the input space. Current systems can process reasonably large inputs, but it is easy to generate formulae that are hard or even impossible to solve. Achieving a

similarly satisfactory situation in the empirical testing of modal decision procedures will require advances in both the decision procedures and the testing methodology.

Acknowledgments

The third author is indebted to Fausto Giunchiglia and Marco Roveri from ITC-IRST, for providing support and feedback, and to Luciano Tubaro from the Maths Department of the University of Trento, for finding out Equation 5.2.

References

[1] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamatiou. Random constraint satisfaction: A more accurate picture. In *ICCP: International Conference on Constraint Programming (CP), LNCS*, 1997.

[2] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics — introduction and summary. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pages 25–26, Berlin, May 1998. Springer-Verlag.

[3] B. Beckert and R. Goré. Free variable tableaux for propositional modal logics. In *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'97*, number 1227 in Lecture Notes in Artificial Intelligence, pages 91–106, Berlin, 1997. Springer-Verlag.

[4] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified Boolean formulae. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 262–267. AAAI Press, 1998.

[5] J. Crawford and L. Auton. Experimental results on the crossover point in satisfiability problems. In *Proc. of the 11th National Conference on Artificial Intelligence*, pages 21–27, 1993.

[6] J. Crawford and L. Auton. Experimental results on the crossover point in random 3-sat. *Artificial Intelligence*, 81(1-2):31–57, 1996.

[7] DIMACS. *The Second DIMACS International Algorithm Implementation Challenge*, Rutgers University, USA, 1993.

[8] I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proc. of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 246–252, Menlo Park, August 4–8 1996. AAAI Press/MIT Press.

[9] I. P. Gent and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Inteligence Research*, 1:47–57, 1993.

[10] I. Gent and T. Walsh. Beyond NP: the QSAT phase transition. In *Proc. of the 16th National Conference on Artificial Intelligence*. AAAI Press, 1999.

[11] I. Gent and T. Walsh. CSPLib: a benchmark library for constraints. Technical report, APES-09-1999, 1999. Available from `http://csplib.cs.strath.ac.uk/`. A shorter version appears in the Proc. of the 5th International Conference on Principles and Practices of Constraint Programming (CP-99).

[12] F. Giunchiglia, M. Roveri, and R. Sebastiani. A new method for testing decision procedures in modal logics. In W. McCune, editor, *Proc. of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 264–267, Berlin, July 13–17 1997. Springer.

[13] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Sixth International Conference (KR'98)*, pages 626–635. Morgan Kaufmann Publishers, San Francisco, California, June 1998.

[14] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. SAT vs. Translation based decision procedures for modal logics: a comparative evaluation. Technical Report 9812-65, IRST, Trento, Italy, December 1998. To appear on *Journal of Applied Non Classical Logics.*

[15] E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT Based Decision Procedures for Classical Modal Logics. *Journal of Automated Reasoning. Special Issue: Satisfiability at the start of the year 2000 (SAT 2000)*, 1999. To appear.

[16] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of the 13th Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag.

[17] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). To appear in "Information and Computation". A longer version is published as ITC-IRST techrep 9611-06, November 1996.

[18] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for ALC. In *Proc. of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, Cambridge, MA, USA, November 1996.

[19] J. Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(3):361–372, 1995.

[20] J. Halpern and Y. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[21] A. Heuerding, G. Jäger, S. Schwendimann, and M. Seyfreid. Propositional logics on the computer. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'95*, number 918 in Lecture Notes in Artificial Intelligence, pages 308–319, Berlin, 1995. Springer-Verlag.

[22] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. Technical Report IAM-96-015, University of Bern, Switzerland, 1996.

[23] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[24] I. Horrocks and P. F. Patel-Schneider. Generating hard modal problems for modal decision procedures. In *Proc. of the First Methods for Modalities Workshop*, May 1999.

[25] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[26] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, volume 1, pages 202–207, 1997.

[27] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4), 1999.

[28] F. Massacci. Design and Results of Tableaux-99 Non-Classical (Modal) System Competition. In *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference (Tableaux'99)*, 1999.

[29] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 459–465, 1992.

[30] P. F. Patel-Schneider. DLP system description. In E. Franconi, G. D. Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 87–89, 1998. Available as CEUR-WS/Vol-11 from `http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS`.

[31] F. J. Pellettier. Seventy-Five Problems for Testing Authomatic Theorem Provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[32] J. Pitt and J. Cunningham. Distributed modal theorem proving with KE. In P. Minglioli, U. Moscato, D. Mindici, and M. Ornaghi, editors, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'96*, number 1071 in Lecture Notes in Artificial Intelligence, pages 160–176, Berlin, 1996. Springer-Verlag.

[33] B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.

[34] C. B. Suttner and G. Sutcliffe. The TPTP Problem Library. Technical Report TR 95/6, James Cook University, Australia, August 1995.

[35] A. Tacchella. *SAT system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proc. of the 1999 International Workshop on Description Logics (DL'99)*, pages 142–144, 1999. Available as CEUR-WS/Vol-22 from `http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS`.

[36] C. P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

Received February 15, 2000

Reachability Logic: An Efficient Fragment of Transitive Closure Logic

NATASHA ALECHINA, *School of Computer Science and IT, University of Nottingham, UK. E-mail: nza@cs.nott.ac.uk*

NEIL IMMERMAN, *Computer Science Dept., UMass, Amherst, USA. E-mail: immerman@cs.umass.edu*

Abstract

We define reachability logic ($\mathcal{RL}$), a fragment of FO^2(TC) (with boolean variables) that admits efficient model checking – linear time with a small constant – as a function of the size of the structure being checked. $\mathcal{RL}$ is expressive enough so that modal logics PDL and CTL* can be linearly embedded in it. The model checking algorithm is also linear in the size of the formula, but exponential in the number of boolean variables occurring in it. In practice this number is very small. In particular, for CTL and PDL formulas the resulting model checking algorithm remains linear. For CTL* the complexity of model checking — which is PSPACE complete in the worst case — can be read from the face of the translated formula.

Keywords: graph query languages, model checking, PDL, CTL*

1 Introduction

Many problems in computer science can be reduced to asking questions about paths in a graph: for example, is there a path from one vertex to another; is the graph connected; is it acyclic; is there a path comprised of a certain kind of steps (e.g. described by a regular expression). We are interested in a logical language that can express these kinds of path queries, or reachability queries, and at the same time where expressions can be efficiently evaluated.

The complexity of query evaluation which we are striving for is linear time in the size of the query times the size of the graph (that is, the number of vertices plus the number of edges in the graph). The graphs under consideration are usually large and sparse.

Several different modal logics have been proposed for talking about paths in a graph, for example, PDL, CTL, CTL*, modal μ-calculus and others. It is not surprising that modal languages are appropriate for expressing reachability queries. They are usually less complex than a 'classical' logic which encompasses them (e.g. basic modal logic vs. first-order logic) but they can quantify over reachable objects and thus express reachability queries.

We believe that a natural 'encompassing' logic for path queries is first-order logic extended with the transitive closure operator [9, 10]. We are looking for a fragment of it which admits efficient model checking and which can express both PDL and CTL*

expressible properties of graphs (note that these two logics are complementary in the sense that neither is more expressive than the other) [5].

In the next section, we formally introduce transitive closure logic and the structures it is interpreted on. Then we introduce a 'modal' fragment $\mathcal{RL}_0$ of $\mathrm{FO}^2(\mathrm{TC})$ where quantifiers are restricted by path descriptions and which admits linear time model checking. However, PDL and CTL* cannot be linearly embedded in it.

In order to express all interesting reachability queries (and provide a linear embedding for PDL and CTL*) $\mathcal{RL}_0$ needs to be extended with additional variables, which range over the set $\{0,1\}$ and are called boolean variables, or booleans. The resulting fragment is called *Reachability Logic* ($\mathcal{RL}$). The model checking algorithm for $\mathcal{RL}$ is linear in the size of the formula and the size of the model, but exponential in the number of booleans in the formula.

We show that PDL and CTL* can be linearly embedded in $\mathcal{RL}$, which results in a new model checking algorithm for these logics. There is no polynomial embedding of CTL* in $\mathcal{RL}_0$, i.e., without the booleans [1]. The booleans are an important indicator of the complexity of a given CTL* query: they make it possible to distinguish 'difficult' queries (which require lots of booleans) from the 'easy' ones just by looking at the syntax of the query.

It can be shown that there is no embedding of PDL in $\mathcal{RL}$ without the booleans. However, embedding PDL uses at most logarithmically many booleans. We show that the model checking algorithm which translates a PDL formula to $\mathcal{RL}$ and evaluates the result is still linear time in the size of the initial PDL formula. The subset CTL of CTL* can be linearly embedded in $\mathcal{RL}_0$ so a linear model checking algorithm results in this case as well.

2 Transitive Closure Logic

The structures of interest to us in this paper are finite labeled graphs, sometimes called Kripke structures. Let $L = \{a, b, \cdots\}$ be a finite set of edge labels and $\Phi = \{p, q, p_1, q_1, \ldots\}$ be a finite set of propositional symbols (vertex labels).

The language $\mathcal{L}(\Phi, L)$ consists of first-order logic with unary relation symbols $\{p : p \in \Phi\}$, binary relation symbols: $\{R_a : a \in L\}$ and equality $=$.

A Kripke structure of vocabulary (Φ, L) is a finite labeled directed graph:

$$\mathcal{K} \quad = \quad \langle S;\ p^{\mathcal{K}} : p \in \Phi;\ R_a^{\mathcal{K}} : a \in L \rangle$$

where S is the set of states (vertices), each $p^{\mathcal{K}} \subseteq S$ is a unary relation on S and each $R_a^{\mathcal{K}} \subseteq S^2$ is a binary relation on S: the set of edges labeled a. Sometimes when it is clear which graph we are talking about we will omit the superscript $^{\mathcal{K}}$.

For any first-order formula φ, we will write $\mathcal{K} \models \varphi$ to mean that φ is true in $\mathcal{K}$.

The following first-order formula says that there is an edge labeled a from the vertex x to some vertex that satisfies p: $\exists y(R_a(x, y) \wedge p(y))$.

Since the reachability relation is not expressible in first-order logic, we cannot yet write such simple formulas as,

"There is a path of a-edges from x to a vertex where p holds." (*)

We next add a transitive closure operator to first-order logic to allow us to express reachability, cf. [9].

Let the formula $\varphi(x_1, \ldots x_k, y_1, \ldots y_k)$ represent a binary relation on k-tuples. We express the reflexive, transitive closure of this relation using the transitive-closure operator (TC), as follows: $\mathrm{TC}_{\overline{x},\overline{y}}\varphi(\overline{x},\overline{y})$ (or $\mathrm{TC}\,\varphi$ if no confusion is likely to arise). Strict transitive closure is denoted by TC^s. Note that strict transitive closure is definable in terms of TC.

Let FO(TC) be the closure of first-order logic under the transitive-closure operator. For example, the following formula expresses (*):

$$\exists y((\mathrm{TC}_{x,y}R_a(x,y))(x,y) \ \wedge \ p(y)).$$

Let $\mathrm{FO}^2(\mathrm{TC})$ be the restriction of first-order logic with transitive closure in which only two variables may appear in a formula (we call them x and y).

3 $\mathcal{RL}_0$, a modal fragment of $FO^2(TC)$

To illustrate the idea behind the reachability logic $\mathcal{RL}$ we first introduce a weaker fragment $\mathcal{RL}_0$. In $\mathcal{RL}_0$, quantifiers are restricted by formulas describing paths. More precisely,

Definition 3.1 *An* adjacency formula *(without booleans) is a quantifier-free formula that is a disjunction of conjunctions where each conjunct contains at least one of $x = y$, $R_a(x,y)$ or $R_a(y,x)$ for some edge label a; in addition, the conjuncts may contain unary atomic formulas of the form $p(x)$.*

Observe that an adjacency formula necessarily implies that there is an edge from x to y or an edge from y to x, or x is equal to y. We allow quantification restricted by an adjacency formula or by a transitive closure of an adjacency formula.

Definition 3.2 *$\mathcal{RL}_0$ is the smallest language that satisfies the following:*

1. *Boolean constants $\top$, $\bot$ are members of $\mathcal{RL}_0$.*
2. *If p is a unary relation symbol, then $p \in \mathcal{RL}_0$.*
3. *If $\varphi, \psi \in \mathcal{RL}_0$, then $\neg\varphi \in \mathcal{RL}_0$ and $\varphi \wedge \psi \in \mathcal{RL}_0$.*
4. *If $\varphi, \psi \in \mathcal{RL}_0$ and q is a new unary predicate symbol, then* (**let** $q = \varphi$ **in** ψ) *is in $\mathcal{RL}_0$.*
5. *If $\varphi \in \mathcal{RL}_0$ and δ is an adjacency formula, then the following formulas are in $\mathcal{RL}_0$:*
   (a) $\mathrm{NEXT}(\delta)\varphi$
   (b) $\mathrm{REACH}(\delta)\varphi$
   (c) $\mathrm{CYCLE}(\delta)$

Semantics of $\mathcal{RL}_0$: *We give the semantics of $\mathcal{RL}_0$ by interpreting each $\mathcal{RL}_0$ construct (on the left) by its meaning in* $\mathrm{FO}^2(\mathrm{TC})$ *on the right. Here the free variable x always refers to the current position.*

$$\begin{aligned} p &\equiv p(x) \\ (\textbf{let } q = \varphi \textbf{ in } \psi) &\equiv \psi[\varphi/q] \end{aligned}$$

$$\begin{aligned} \mathrm{NEXT}(\delta)\varphi &\equiv \exists y(\delta(x,y) \wedge \varphi[y/x]) \\ \mathrm{REACH}(\delta)\varphi &\equiv \exists y(\mathrm{TC}\,\delta)(x,y) \wedge \varphi[y/x]) \\ \mathrm{CYCLE}(\delta) &\equiv (\mathrm{TC}^s\,\delta)(x,x) \end{aligned}$$

The idea behind the logic $\mathcal{RL}_0$ is that we may speak about the current vertex (x), steps out of x, paths out of x, and cycles from x back to itself. $\mathcal{RL}_0$ may be thought of as a modal logic, or as a fragment of $\mathrm{FO}^2(\mathrm{TC})$ in which only the variable x may occur free. Clearly $\mathcal{RL}_0$ is a proper fragment of $\mathrm{FO}^2(\mathrm{TC})$ because $\mathcal{RL}_0$ may not discuss vertices unreachable from the current vertex. For example, the following formula is not expressible in $\mathcal{RL}_0$,

$$\forall y(\mathrm{TC}\,R_a)(x,y)\ .$$

The **let** construct allows for simpler and more modular formulas. It allows us to use a new symbol q for a formula φ and thus may save space in a formula in which we had to write out φ several times. Furthermore, without the **let** construct we could not restrict adjacency formulas to be quantifier free.

The following are a few sample $\mathcal{RL}_0$ formulas and their meanings. Note that the third formula asserts that there exists an infinite chain. Thus $\mathcal{RL}_0$ does not have the finite model property. However, we do restrict our attention in this paper to finite structures.

1. $\neg\mathrm{REACH}(R(x,y))\mathrm{CYCLE}(R(x,y) \wedge p(x) \wedge \neg q(x))$ means that there is no reachable cycle along which p is always true and q is always false. Over finite structures this expresses weak fairness, i.e, there is no infinite path along which a resource is always requested (p) but never granted (q).
2. $\neg\mathrm{REACH}(R(x,y))(p \wedge \mathrm{CYCLE}(R(x,y) \wedge \neg q(x)))$ means that there is no reachable cycle along which p occurs at least once but q never holds. Over finite structures this expresses strong fairness, i.e., there is no infinite path along which a request is made infinitely often but granted only finitely often.
3. $\neg\mathrm{REACH}(R(x,y))(\neg\mathrm{NEXT}(R(x,y))\top \ \vee \ \mathrm{CYCLE}(R(x,y)))$ means that every reachable point has a successor and is not involved in a cycle.
4. **let** $r = \mathrm{REACH}(R_a(x,y))p$ **in** $\mathrm{REACH}(R(x,y) \wedge r)\mathrm{CYCLE}(R(x,y) \wedge r)$ means there is an infinite path for which at all times we can take a path of a-edges to a point where p holds.

$\mathcal{RL}_0$ is unexpectedly similar to other 'bounded quantifier fragments' (see [3]). However, instead of quantifying over objects accessible by an atomic step, in $\mathcal{RL}_0$ we quantify over objects reachable by a path. The reason for restricting quantifiers in $\mathcal{RL}_0$ is not the quest for decidability, as in bounded fragments, but for efficient evaluation. The following theorem illustrates this point.

Theorem 3.3 *There is an algorithm that given a graph G and a formula $\varphi \in \mathcal{RL}_0$ marks the vertices in G that satisfy φ. This algorithm runs in time $O(|G||\varphi|)$.*

Proof. We inductively mark the vertices of G according to whether they satisfy each subformula of φ.

We assume that G is represented by a sorted adjacency list. That is, for each vertex s we have the list of all adjacent vertices, $v_1, \ldots, v_d$. For each v_i on the list we have a

label indicating exactly which edge relations hold, e.g. $R_a(s, v_1)$ and $R_b(v_1, s)$. This structure is linear in the size of G (number of vertices + double the number of edges).

We also assume that we have a list of all subformulas of φ in the order of increasing complexity. We iterate through the list, and for every vertex s we mark it according to the following rules:

1. **Base case:** Mark vertex s for p, iff $s \in p^G$.
2. Mark s for $\neg\varphi$ iff s is not marked for φ. Mark s for $\varphi \wedge \psi$ iff s is marked for φ and s is marked for ψ.
3. There are three cases in the remaining part of the marking algorithm:
   (a) Mark s for $\mathrm{NEXT}(\delta(x, y))\varphi$ iff there is an edge from s to a vertex s' that is marked for φ and such that $\delta(s, s')$ holds. Note that the time required to check all adjacent vertices for every vertex is at most the size of δ times the size of the adjacency list.
   (b) Mark s for $\mathrm{REACH}(\delta)\varphi$ iff there is a δ-path to some vertex s' that is marked for φ. This can be tested in linear time by doing a depth first search of G starting from all points marked for φ and proceeding backwards along edges for which δ holds.
   (c) Mark s for $\mathrm{CYCLE}(\delta)(x, x)$ iff s is in a non-trivial strongly connected component of the δ-graph. This can also be checked in linear time using depth first search of G [2]. ∎

Although we feel that $\mathcal{RL}_0$ is an interesting fragment of $\mathrm{FO}^2(\mathrm{TC})$ in itself, and it admits linear time model checking, we need to show that $\mathcal{RL}_0$ can express interesting queries. Unfortunately, we cannot show this by (linearly) embedding modal logics PDL and CTL* in $\mathcal{RL}_0$. Indeed, we next show that PDL is embeddable neither in $\mathcal{RL}_0$ nor in full $\mathrm{FO}^2(\mathrm{TC})$ without booleans.

Proposition 3.4 *PDL cannot be embedded in $\mathcal{RL}_0$, nor in $\mathrm{FO}^2(\mathrm{TC})$ without booleans.*

Proof. Consider the PDL formula $\mathrm{EVEN} \equiv \langle (a; a)^\star \rangle \neg \langle a \rangle \top$ meaning that there is an even length path of a's from where we arrive to a point that has no a-edge going out of it. If $\mathrm{FO}^2(\mathrm{TC})$ without booleans is interpreted over finite successor structures, then every $\mathrm{FO}^2(\mathrm{TC})$ formula is equivalent to a two-variable first-order formula with order. (Note that no such formula can express the property that the distance from x to y is two.) But over finite orderings, EVEN is not expressible in first-order logic[1]. ∎

4 Reachability Logic

In order to provide linear embeddings of PDL and CTL* in $\mathrm{FO}^2(\mathrm{TC})$ we need to introduce additional expressive power. In addition to ordinary (or domain variables, which range over the domain of the input structure, we allow *boolean variables* $b, c, d, b_1, \ldots$. Boolean variables are essentially first-order variables that are restricted to range only over the first two elements of the universe, which we fix as 0 and 1.

In what follows, we will assume that $\mathrm{FO}^2(\mathrm{TC})$ may contain boolean variables.

We modify the definition of an adjacency formula as follows:

[1] We thank the anonymous referee who suggested this simpler version of our original proof.

Definition 4.1 *An* adjacency formula *(with booleans) is a disjunction of conjunctions where each conjunct contains at least one of $x = y$, $R_a(x, y)$ or $R_a(y, x)$ for some edge label a; in addition, the conjuncts may contain expressions of the form $(\neg)(b_1 = b_2)$, $(b_1 = 0)$, $(b_1 = 1)$ and $p(x)$, where b_1 and b_2 are boolean variables.*

Definition 4.2 *$\mathcal{RL}$ is the smallest fragment of* $\mathrm{FO}^2(\mathrm{TC})$ *that satisfies the following:*

1. *If p is a unary relation symbol then $p \in \mathcal{RL}$; also $\top, \bot \in \mathcal{RL}$.*
2. *If $\varphi, \psi \in \mathcal{RL}$, then $\neg\varphi \in \mathcal{RL}$ and $\varphi \wedge \psi \in \mathcal{RL}$.*
3. *If $\varphi \in \mathcal{RL}$ and b is a boolean variable, then $\exists b\varphi \in \mathcal{RL}$.*
4. *If $\varphi, \psi \in \mathcal{RL}$ and q is a new unary predicate symbol, then (**let** $q = \varphi$ **in** ψ) is in $\mathcal{RL}$.*
5. *If $\varphi \in \mathcal{RL}$ and $\delta(x, \overline{b}, y, \overline{b'})$ is an adjacency formula (a binary relation between two n-tuples $\langle x, b_1, \ldots, b_{n-1}\rangle$ and $\langle y, b'_1, \ldots, b'_{n-1}\rangle$), then the following formulas are in $\mathcal{RL}$:*
   (a) $\mathrm{NEXT}(\delta)\varphi$
   (b) $\mathrm{REACH}(\delta)\varphi$
   (c) $\mathrm{CYCLE}(\delta)$

Semantics of $\mathcal{RL}$: *The semantics of $\mathcal{RL}$ is similar to that of $\mathcal{RL}_0$, the only difference being the use of booleans in adjacency formulas. In each case below assume that $\delta(x, \overline{b}, y, \overline{b'})$ is an adjacency formula.*

$$\begin{aligned} p &\equiv p(x) \\ (\textbf{let } q = \varphi \textbf{ in } \psi) &\equiv \psi[\varphi/q] \\ \mathrm{NEXT}(\delta)\varphi &\equiv \exists y(\delta(x, \overline{0}, y, \overline{1}) \wedge \varphi[y/x]) \\ \mathrm{REACH}(\delta)\varphi &\equiv \exists y(\mathrm{TC}\,\delta)(x, \overline{0}, y, \overline{1}) \wedge \varphi[y/x]) \\ \mathrm{CYCLE}(\delta) &\equiv (\mathrm{TC}^s\,\delta)(x, \overline{0}, x, \overline{0}) \end{aligned}$$

Here are some examples of formulas in $\mathcal{RL}$:

- $\mathrm{REACH}(\delta)p$ where $\delta(x, b_1, b_2, y, b'_1, b'_2)$ is $(R_a(x, y) \wedge b_1b_2 = 00 \wedge b'_1b'_2 = 01) \vee (R_b(x, y) \wedge b_1b_2 = 01 \wedge b'_1b'_2 = 11)$ (this is $\langle a; b\rangle p$ of PDL, see Section 6).
- $\varphi_1 = \mathrm{REACH}(R)p$ (**EF**p of CTL*, see Section 5).
- $\varphi_2 = \mathrm{REACH}(\delta)\mathrm{CYCLE}(\delta)$, where δ is $R(x, y) \wedge q(x)$ (**EG**q of CTL*).
- (**let** $q = \varphi_1$ *in* φ_2) (**EGEF**p of CTL*).

$\mathcal{RL}$ is a logical language and it is a fragment of $\mathrm{FO}^2(\mathrm{TC})$. However, because of the 'let' construct, when we talk about size in the representation of $\mathcal{RL}$, we are really talking about circuits. Thus the size of an $\mathcal{RL}$-circuit may be logarithmic in the size of the smallest equivalent $\mathrm{FO}^2(\mathrm{TC})$ formula. This allows the linear size embedding of CTL* which presumably does not hold for $\mathrm{FO}^2(\mathrm{TC})$ (without a circuit representation or an extra domain variable cf. [10]).

Boolean variables however add extra complexity, which is not surprising since model checking CTL* is PSPACE complete [13].

Theorem 4.3 *There is an algorithm that given a graph G and a formula $\varphi(x) \in \mathcal{RL}$ marks the vertices in G that satisfy φ. This algorithm runs in time $O(|G||\varphi|2^{n_b})$ where n_b is the number of boolean variables occurring in φ.*

PROOF. As for $\mathcal{RL}_0$, we inductively mark the vertices of G according to whether they satisfy each subformula of φ. For subformulas that include a free boolean variable, we include this subformula with both substitutions of the boolean variable, thus with n_b booleans there could be as many as 2^{n_b} copies of some subformulas.

In $\mathcal{RL}$, formulas talk about transitions between tuples of the form (node, sequence of booleans). In addition to G, we need to maintain an adjacency list corresponding to the extended graph $\mathcal{G}$, where nodes correspond to old nodes followed by sequences of 0s and 1s of length n_b. $\mathcal{G}$ is exponentially (in the number of booleans) larger than G.

We also assume that we have a list of all subformulas of φ in the order of increasing complexity. We iterate through the list, and for every vertex s mark it as we did for $\mathcal{RL}_0$. The new cases are:

- Mark s for $(\exists b)\varphi$ iff s is marked for at least one of $\varphi(0/b)$ or $\varphi(1/b)$.
- For the formula (**let** $q = \varphi$ **in** ψ), we have inductively marked states according to whether they satisfy φ. Thus, our Kripke structure is expanded to interpret the new predicate symbol q, true of those states marked for φ. Now we evaluate the smaller formula ψ on this expanded structure.
- The cases of NEXT(δ)φ, REACH(δ)φ, and CYCLE(δ) are the same as for $\mathcal{RL}_0$, but we do depth first search of $\mathcal{G}$ instead of G. For the last clause, we check that $s, \overline{0}$ is in a non-trivial strongly connected component of the δ-subgraph of $\mathcal{G}$.

It is easy to see that the above marking algorithm is correct and runs in the required time. ∎

5 Embedding CTL* in $\mathcal{RL}$

A popular and quite expressive language for Model Checking is computation tree logic CTL*. CTL* is a version of temporal logic that combines linear and branching time. CTL* has two kinds of formulas: *state formulas*, which are true or false at each state, and *path formulas*, which are true or false with respect to an infinite path through the model. The following is an inductive definition of the state and path formulas of CTL*.

Definition 5.1 (Syntax of CTL*) *State formulas $\mathcal{S}$ and path formula $\mathcal{P}$ of* CTL* *are the smallest sets of formulas satisfying the following:*

State Formulas, $\mathcal{S}$:
the boolean constants $\top$ and $\bot$ are elements of $\mathcal{S}$;
if $p_i \in \Phi$, then $p_i \in \mathcal{S}$;
if $\varphi \in \mathcal{P}$, then $\mathbf{E}\varphi \in \mathcal{S}$.

Intuitively, $\mathbf{E}\varphi$ means that there exists an infinite path starting at the current state and satisfying φ.

Path Formulas, $\mathcal{P}$:

if $\alpha \in \mathcal{S}$ *then* $\alpha \in \mathcal{P}$*;*
if $\varphi, \psi \in \mathcal{P}$*, then* $\neg\varphi, \varphi \wedge \psi$*,* $\mathbf{X}\varphi$*, and* $\varphi\mathbf{U}\psi$ *are in* $\mathcal{P}$*.*

Intuitively, $\mathbf{X}\varphi$ *means that* φ *holds at the next moment of time and* $\varphi\mathbf{U}\psi$ *means that at some time now or in the future,* ψ *holds, and from now until then,* φ *holds.*

Next, we formally define the semantics of the above operators. A path $\rho = \rho_0, \rho_1, \ldots$ is a mapping of the natural numbers to states in $\mathcal{K}$ such that for all i, $\mathcal{K} \models R(\rho_i, \rho_{i+1})$. We use the notation ρ^i for the tail of ρ, with states $\rho_0, \rho_1, \ldots, \rho_{i-1}$ removed.

Definition 5.2 (Semantics of CTL***)** *The following are inductive definitions of the meaning of* CTL* *formulas:*

State Formulas:

$$\begin{array}{rcl} (\mathcal{K}, s) \models p_i & \textit{iff} & \mathcal{K} \models p_i(s) \\ (\mathcal{K}, s) \models \mathbf{E}\varphi & \textit{iff} & \exists \textit{ path } \rho(\rho_0 = s \wedge (\mathcal{K}, \rho) \models \varphi) \end{array}$$

Path Formulas:

$$\begin{array}{rcl} (\mathcal{K}, \rho) \models \alpha & \textit{iff} & (\mathcal{K}, \rho_0) \models \alpha \textit{ for } \alpha \in \mathcal{S} \\ (\mathcal{K}, \rho) \models \varphi \wedge \psi & \textit{iff} & (\mathcal{K}, \rho) \models \varphi \textit{ and } (\mathcal{K}, \rho) \models \psi \\ (\mathcal{K}, \rho) \models \neg\varphi & \textit{iff} & (\mathcal{K}, \rho) \not\models \varphi \\ (\mathcal{K}, \rho) \models \mathbf{X}\varphi & \textit{iff} & (\mathcal{K}, \rho^1) \models \varphi \\ (\mathcal{K}, \rho) \models \varphi\mathbf{U}\psi & \textit{iff} & \exists i((\mathcal{K}, \rho^i) \models \psi \wedge (\forall j < i)(\mathcal{K}, \rho^j) \models \varphi) \end{array}$$

Theorem 5.3 *There is a linear-time computable function* g *that maps any* CTL* *formula* φ *to an equivalent formula* $g(\varphi) \in \mathcal{RL}$*. While* $g(\varphi)$ *has only two domain variables, it may have a linear number of boolean variables.*

Proof. We review the proof from [10] that CTL* is linearly embeddable in FO^2(TC) and show that the embedding lands in $\mathcal{RL}$.

Let $\mathbf{E}(\varphi)$ be a CTL* formula in which the "$\neg$"s have been pushed inside as far as possible subject to the fact that all path quantifiers should be **E**s. For this purpose we will need the temporal operator **B**, the dual of **U**,

$$\varphi\mathbf{B}\psi \quad \equiv \quad \neg(\neg\varphi\mathbf{U}\neg\psi)$$

The intuitive meaning of $\varphi\mathbf{B}\psi$ is that "φ holds before ψ fails."[2]

Inductively assume that we have computed $g(\alpha)$ for every state subformula of φ. We can then use the "let" rule of $\mathcal{RL}$ to replace $g(\alpha)$ by a new unary relation symbol. We can thus assume that φ has no path quantifiers.

Define the *closure* of φ ($cl(\varphi)$) to be the set of all subformulas of φ. We introduce a boolean variable b_α for each $\alpha \in cl(\varphi)$. Intuitively, we use the boolean variables to encode the state of the automaton that runs along a path and checks that the path

[2] Other authors use "**R**" for the dual of **U** and say that, "φ releases ψ," (from the obligation of holding in the future).

satisfies a path formula (see [14]). We do not need booleans for state formulas but we use them just to make the following inductive definition simpler.

Let $\overline{b}$ be a tuple of all the boolean variables b_α, for $\alpha \in cl(\varphi)$. Define the transition relation $\delta^0_\varphi(y, \bar{b}, y', \overline{b'})$ as follows. In each case, the comment on the right is the condition under which the given conjunct is included in the formula. (We assume that φ is written in positive-normal form.)

	$R(x,y)$	
$\wedge$	$b_\alpha = 1 \ \rightarrow g(\alpha)(x)$	$\forall$ state formula $\alpha \in cl(\varphi)$
$\wedge$	$b_{\alpha\wedge\beta} = 1 \ \rightarrow b_\alpha = 1 \wedge b_\beta = 1$	$\forall$ path formula $\alpha \wedge \beta \in cl(\varphi)$
$\wedge$	$b_{\alpha\vee\beta} = 1 \ \rightarrow b_\alpha = 1 \vee b_\beta = 1$	$\forall$ path formula $\alpha \vee \beta \in cl(\varphi)$
$\wedge$	$b_{\mathbf{X}\alpha} = 1 \ \rightarrow b'_\alpha = 1$	$\forall$ path formula $\mathbf{X}\alpha \in cl(\varphi)$
$\wedge$	$b_{\alpha\mathbf{U}\beta} = 1 \ \rightarrow b_\beta = 1 \vee (b_\alpha = 1 \wedge b'_{\alpha\mathbf{U}\beta} = 1)$	$\forall$ path formula $\alpha\mathbf{U}\beta \in cl(\varphi)$
$\wedge$	$b_{\alpha\mathbf{B}\beta} = 1 \ \rightarrow b_\beta = 1 \wedge (b_\alpha = 1 \vee b'_{\alpha\mathbf{B}\beta} = 1)$	$\forall$ path formula $\alpha\mathbf{B}\beta \in cl(\varphi)$

It follows that if $b_\varphi = 1$, then an infinite δ^0_φ-path starting at $(\overline{b}, x)$ may satisfy φ. However, there could be some booleans $b_{\alpha\mathbf{U}\beta}$ that are true, promising that eventually β will become true, but in fact as we walk around a cycle, α remains true but β never becomes true.

In order to solve this problem, let $\overline{m}$ be a tuple of bits m_β, one for each "Until" formula, $\alpha\mathbf{U}\beta \in cl(\varphi)$. We use the "memory bit" m_β to check that β actually occurs by starting it at 0 and only letting it become 1 when β becomes true.

Let $\overline{d}$ be a set of $|cl(\varphi)|$ "destination" bits and let c_0, c_1 be "control" bits. Define the adjacency formulas δ_1 and δ_2 as follows. They imply that δ_1 starts with $b_\varphi = 1$ and ends at $\overline{b} = \overline{d}$. Similarly δ_2 starts with $\overline{b} = \overline{d}$ and the memory bits all zero and ends with $\overline{b} = \overline{d}$ and the memory bits all one:

$$\begin{aligned}
\delta_1(\overline{c}, \overline{b}, x, \overline{c'}, \overline{b'}) \quad \equiv \quad & (\overline{c} = 00 \wedge \overline{c'} = 01 \wedge b'_\varphi = 1 \wedge x = y) \\
\vee \quad & (\overline{c} = \overline{c'} = 01 \wedge \delta^0_\varphi(\overline{b}, x, \overline{b'}, y)) \\
\vee \quad & (\overline{c} = 01 \wedge \overline{c'} = 11 \wedge \overline{b} = \overline{d} \wedge \overline{b'} = \overline{1} \wedge x = y)
\end{aligned}$$

$$\begin{aligned}
\delta_2(\overline{c}, \overline{b}, \overline{m}, x, \overline{c'}, \overline{b'}, \overline{m'}) \quad \equiv \quad & (\overline{c} = 00 \wedge \overline{c'} = 01 \wedge \overline{b'} = \overline{d} \wedge \overline{m} = \overline{0} \wedge x = y) \\
\vee \quad & (\overline{c} = \overline{c'} = 01 \wedge \delta^0_\varphi(\overline{b}, x, \overline{b'}, y) \\
& \quad \wedge (m'_\beta = 1 \rightarrow (m_\beta = 1 \vee b_\beta = 1)) \quad \text{for } \alpha\mathbf{U}\beta \in cl(\varphi)\ ) \\
\vee \quad & (\overline{c} = 01 \wedge \overline{b} = \overline{d} \wedge \overline{c'} = \overline{0} \wedge \overline{m'} = \overline{b'} = \overline{0} \wedge x = y \\
& \quad \wedge (b_{\alpha\mathbf{U}\beta} = 1 \rightarrow m_\beta = 1) \quad \text{for any } \alpha\mathbf{U}\beta \in cl(\varphi)\ )
\end{aligned}$$

We define the desired mapping g from CTL* state formulas to $\mathcal{RL}$ as follows:

$$g(\mathbf{E}\varphi) \equiv \exists\overline{d}\mathrm{REACH}(\delta_1)\mathrm{CYCLE}(\delta_2)$$

By construction, $g(\mathbf{E}\varphi)$ asserts that there is an infinite path along which φ holds, as desired. ∎

6 Embedding PDL in $\mathcal{RL}$

Propositional Dynamic Logic (PDL) was introduced in [12] as a logic to reason about programs. The language of PDL includes two sets of primitive symbols: a set of propositional symbols and a set of atomic transitions. Propositional symbols stand for properties that can be true or false for a node in a graph (in the original interpretation of PDL, they are properties of states in the execution of a program). Atomic transitions (edge labels) are interpreted in PDL as basic instructions, e.g., assignment statements.

Definition 6.1 (Syntax of PDL) **Transition terms, $\mathcal{T}$:**

elements of L (edge labels) are transition terms;
if $t \in \mathcal{T}$, then $t^* \in \mathcal{T}$;
if $t_1, t_2 \in \mathcal{T}$, then $t_1; t_2$ and $t_1 \cup t_2$ are in $\mathcal{T}$;
if φ is a formula, then $\varphi? \in \mathcal{T}$.

Intuitively, ';' corresponds to sequential composition, '$\cup$' to non-deterministic choice, '$*$' to finite iteration of unspecified length and '$\varphi?$' to test for φ.

Formulas, $\mathcal{F}$:

the boolean constants $\top$ and $\bot$ are elements of $\mathcal{F}$;
if $p_i \in \Phi$, then $p_i \in \mathcal{F}$;
if $\varphi, \psi \in \mathcal{F}$ and $t \in \mathcal{T}$, then $\neg\varphi, \varphi \wedge \psi$, $\langle t \rangle \varphi$ are in $\mathcal{F}$.

The formula $\langle t \rangle \varphi$ means 'after some transition t, φ holds' .

For example, $\langle a^* \rangle \neg \langle b \rangle \top$ means that after 0 or finitely many a links, one can reach a node that has no outgoing links labeled b.

The language of PDL is interpreted by Kripke structures.

Definition 6.2 (Semantics of PDL) *The meaning of transition terms is given by the following function tr:*

$tr(a) = R_a$
$tr(t_1 \cup t_2) = tr(t_1) \cup tr(t_2)$
$tr(t*)$ *is the reflexive, transitive closure of* $tr(t)$
$tr(t_1; t_2) = \{(u, v) : \exists z (tr(t_1)(u, z) \wedge tr(t_2)(z, v))\}$
$tr(\varphi?) = \{(u, u) : \varphi$ *is true at* $u\}$

The following are inductive definitions of the meaning of PDL formulas:

$$
\begin{aligned}
(\mathcal{K}, s) \models p_i \quad & \textit{iff} \quad \mathcal{K} \models p_i(s) \\
(\mathcal{K}, s) \models \varphi \wedge \psi \quad & \textit{iff} \quad (\mathcal{K}, s) \models \varphi \textit{ and } (\mathcal{K}, s) \models \psi \\
(\mathcal{K}, s) \models \neg\varphi \quad & \textit{iff} \quad (\mathcal{K}, s) \not\models \varphi \\
(\mathcal{K}, s) \models \langle t \rangle \varphi \quad & \textit{iff} \quad \exists s' ((s, s') \in tr(t) \textit{ and } (\mathcal{K}, s') \models \varphi)
\end{aligned}
$$

PDL model checking is linear time, i.e., $O(|K||\varphi|)$. This follows from the fact that PDL can be linearly embedded into alternation-free μ-calculus circuits [5], and the latter can be model checked in linear time [4]. In this section we show how to linearly embed PDL into a portion of $\mathcal{RL}$ which admits linear time model checking.

We begin by informally showing how to model check PDL in linear time and then how to preserve this linear-time algorithm as we first map PDL to $\mathcal{RL}$ and then model-check the resulting formula.

We are given a PDL formula φ, and a Kripke structure, $\mathcal{K}$. We want to mark each state of $\mathcal{K}$ according to whether it satisfies each subformula of φ.

The only tricky case is $\langle\alpha\rangle\psi$, where we assume that we have inductively marked all states satisfying ψ. The formula α is a regular expression which can be translated to an nondeterministic finite automata N_α of size $|N_\alpha| = O(|\alpha|)$.

Next, we can mark all states of $\mathcal{K}$ such that a string in $\mathcal{L}(N_\alpha)$ can take them to a state marked ψ. This is just a depth-first search of $\mathcal{K} \times N_\alpha$, taking time $O(|K||N|)$[3]. The desired time $O(|K||\varphi|)$ model checking algorithm for PDL results.

The above linear-time model checking algorithm for PDL suggests a natural way to translate PDL to $\mathcal{RL}$. Define a linear translation h from PDL to $\mathcal{RL}$ as follows:

$$\begin{array}{rcl} h(p) & = & p \\ h(\alpha \wedge \beta) & = & h(\alpha) \wedge h(\beta) \\ h(\neg\alpha) & = & \neg h(\alpha) \\ h(\langle\alpha\rangle\psi) & = & \mathrm{REACH}(\delta_\alpha)h(\psi) \end{array}$$

The only interesting case in the above definition of h is the last. Here, δ_α is a translation of the transition relation of N_α using $\log|\alpha| + O(1)$ pairs of booleans to encode the state. Clearly δ_α can be written in disjunctive normal form in size and time that is linear in the size of N_α[4]. Each clause of δ_α is of the form

$$\bar{a} = \bar{c} \ \wedge \ \overline{a'} = \bar{d} \wedge R_a(x, y)$$

where N_α has a transition from state $\bar{c}$ to state $\bar{d}$ reading the letter a. We encode the start state as $\bar{0}$ a tuple of zero's and similarly we can have N_α have a unique final state and encode it as $\bar{1}$.

It is easy to see that,

Lemma 6.3 *The mapping h from PDL to $\mathcal{RL}$ defined above is computable in linear time. Furthermore, for all PDL formulas φ, and Kripke structures $\mathcal{K}$ with states s, we have that,*

$$\mathcal{K}, s \models \varphi \quad \Leftrightarrow \quad \mathcal{K}, s \models h(\varphi)$$

It is intuitively clear that the resulting formula $h(\varphi) \in \mathcal{RL}$ can be model checked in linear time.

Lemma 6.4 *Model checking for formulas in the image of PDL under the mapping h is linear.*

Proof. Let $\varphi \in \mathcal{RL}$ be a translation of a PDL formula. Observe that all adjacency formulas δ in φ are in complete DNF, meaning that for each clause t, and each boolean variable b, one of $b = 0$ or $b = 1$ occurs in t.

---

[3] Note that if α includes a subformula of the form "$\gamma?$" then inductively we have marked the states of $\mathcal{K}$ according to whether or not they satisfy γ. For a state that satisfies γ, $\gamma?$ is a possible move that leaves the state of $\mathcal{K}$ fixed, whereas for a state that does not satisfy γ, $\gamma?$ is not possible.

[4] Here we are assuming that the size of an equation of the form $\bar{a} = \bar{c}$ is $O(1)$ where $\bar{a}$ and $\bar{c}$ are $\log n$-tuples of booleans. This is consistent with the unit cost for operations on $\log n$-bit words that underly most linear-time algorithms [2].

The model checking algorithm is simpler than the algorithm for the full $\mathcal{RL}$ with no free boolean variables. Again the only interesting case is (5). We know that for all formulas which are translations of PDL formulas under h, there are no occurrences of NEXT or CYCLE, and all occurrences of REACH are of the form

$$\varphi \equiv \text{REACH}(\delta_\alpha)h(\psi)$$

We may assume that we have inductively marked the states satisfying $h(\psi)$. As in the model checking algorithm for $\mathcal{RL}$, we replace the graph G with a larger graph $\mathcal{G}$ where each node is an original node of G followed by a sequence of 0s and 1s. Note that $\mathcal{G} = G \times N_\alpha$ is of size at most $O(|G||\delta|)$. The linear-time model checking algorithm results. ∎

7 Conclusions

We have shown that PDL and CTL* can be embedded in reachability logic, $\mathcal{RL} \subseteq \text{FO}^2(\text{TC})$. $\mathcal{RL}$ can be efficiently model checked in time $O(|K||\varphi|2^{n_b})$: linear in the size of the structure and the formula, but exponential in the number of booleans in the formula. Furthermore, for PDL and CTL this algorithm is linear, i.e., $O(|K||\varphi|)$.

This is useful, both because n_b tends to be tiny, and because the language involved is closely tied to reachability queries which are the bread and butter of model checking. One nice feature is that we can look at the formula, count the number of booleans, and automatically say whether the query can be checked efficiently or not. (Recall that model checking CTL* is PSPACE complete [13]. Thus there presumably are some CTL* queries that are not feasible. An advantage of translating to $\mathcal{RL}$ is that we can see whether or not it is feasible on the face of the resulting query.)

We believe that model checking using $\mathcal{RL}$ may be more efficient than using the μ-calculus in many practical cases. The following directions should be investigated concerning model checking via transitive closure logic:

- Dynamic model checking strategies are needed, i.e., we should often be able to efficiently recompute a model checking query after a small change in the design being checked.
- We have only shown that explicit model checking for $\mathcal{RL}$ is efficient. We believe the same will be true of symbolic model checking but this needs to be investigated.
- From Savitch's theorem, we know that reachability is contained in DSPACE[$\log^2 n$]. The time/space tradeoff for computing reachability should be investigated and exploited in model checking.

Acknowledgment

This research was supported in part by NSF grant CCRR-9877078.

References

[1] M. Adler and N. Immerman. An $n!$ lower bound on formula size. Manuscript, 1999.

[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] H. Andréka, J. van Benthem, and I. Németi, Modal Languages and Bounded Fragments of Predicate Logic. *J. Philos. Logic*, 27(3):217–274, 1998.

[4] R. Cleaveland and B. Steffen. A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal μ-Calculus. *Formal Methods in System Design 2*, 121-147, 1993.

[5] E.A. Emerson and C.-L. Lei. Efficient Model Checking in Fragments of the Propositional *mu*-Calculus. In *Proc. LICS'86*, 267-278, 1986.

[6] K. Etessami and N. Immerman. Tree Canonization and Transitive Closure. To appear in *Information and Computation*. A preliminary version appeared in *Proc. LICS'95*, 331-341, 1995.

[7] K. Etessami and T. Wilke. An Until Hierarchy for Temporal Logic. In *Proc. LICS'96*, 1996.

[8] N. Immerman. *Descriptive Complexity*, Springer Graduate Texts in Computer Science, New York, 1998.

[9] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16(4):760–778, 1987.

[10] N. Immerman and M. Vardi, Model Checking and Transitive Closure Logic. *Proc. 9th Int. Conf. on Computer-Aided Verification* (CAV'97), Lecture Notes in Computer Science, Springer-Verlag 291 - 302, 1997.

[11] N. Immerman, DSPACE[n^k]=VAR[$k+1$]. *Sixth IEEE Structure in Complexity Theory Symposium*, 334-340, 1991.

[12] V. R. Pratt, Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[13] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. In *JACM,* 32(3):733-749, 1985.

[14] M. Y. Vardi and P. Wolper. Reasoning about Infinite Computations. In *Information and Computation*, 115(1):1-37, 1994.

Received February 14, 2000

Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto

PATRICK BLACKBURN, *Computerlinguistik, Universität des Saarlandes, 66041 Saarbrücken, Germany.*
E-mail: patrick@coli.uni-sb.de

Abstract

This paper is about the good side of modal logic, the bad side of modal logic, and how hybrid logic takes the good and fixes the bad.

In essence, modal logic is a simple formalism for working with relational structures (or multigraphs). But modal logic has no mechanism for referring to or reasoning about the individual nodes in such structures, and this lessens its effectiveness as a representation formalism. In their simplest form, hybrid logics are upgraded modal logics in which reference to individual nodes is possible.

But hybrid logic is a rather unusual modal upgrade. It pushes one simple idea as far as it will go: represent *all* information as formulas. This turns out to be the key needed to draw together a surprisingly diverse range of work (for example, feature logic, description logic and labelled deduction). Moreover, it displays a number of knowledge representation issues in a new light, notably the importance of sorting.

Keywords: Labelled deduction, description logic, feature logic, hybrid logic, modal logic, sorted modal logic, temporal logic, nominals, knowledge representation, relational structures

1 Modal Logic and Relational Structures

To get the ball rolling, let's recall the syntax and semantics of (propositional) multimodal logic.

Definition 1.1 (Multimodal languages) *Given a set of* propositional symbols *$PROP = \{p, q, p', q', \dots\}$, and a set of modality labels $MOD = \{\pi, \pi', \dots\}$, the set of well-formed formulas of the multimodal language (over PROP and MOD) is defined as follows:*

$$WFF := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \langle\pi\rangle\varphi \mid [\pi]\varphi,$$

for all $p \in PROP$ and $\pi \in MOD$. As usual, $\varphi \leftrightarrow \psi \ := \ (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

Definition 1.2 ((Kripke) models) *Such a language is interpreted on* models *(often called* Kripke models*). A model $\mathcal{M}$ (for a fixed choice of PROP and MOD) is a triple $(W, \{R_\pi \mid \pi \in MOD\}, V)$. Here W is a non-empty set (I'll call its elements states, or nodes), and each R_π is a binary relations on W. The pair $(W, \{R_\pi \mid \pi \in MOD\})$ is called the* frame *underlying $\mathcal{M}$, and $\mathcal{M}$ is said to be a model based on this frame. V (the* valuation*) is a function with domain PROP and range $Pow(W)$; it tells us at which states (if any) each propositional symbol is true.*

Definition 1.3 (Satisfaction and validity) *Interpretation is carried out using the Kripke satisfaction definition. Let* $\mathcal{M} = (W, \{R_\pi \mid \pi \in MOD\}, V)$ *and* $w \in W$*. Then:*

$$\begin{array}{lll}
\mathcal{M}, w \Vdash p & \textit{iff} & w \in V(p), \ \textit{where } p \in \mathrm{PROP} \\
\mathcal{M}, w \Vdash \neg\varphi & \textit{iff} & \mathcal{M}, w \nVdash \varphi \\
\mathcal{M}, w \Vdash \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \Vdash \varphi \textit{ and } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \vee \psi & \textit{iff} & \mathcal{M}, w \Vdash \varphi \textit{ or } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \rightarrow \psi & \textit{iff} & \mathcal{M}, w \nVdash \varphi \textit{ or } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \langle\pi\rangle\varphi & \textit{iff} & \exists w'(wR_\pi w' \ \&\ \mathcal{M}, w' \Vdash \varphi) \\
\mathcal{M}, w \Vdash [\pi]\varphi & \textit{iff} & \forall w'(wR_\pi w' \ \Rightarrow \ \mathcal{M}, w' \Vdash \varphi).
\end{array}$$

If $\mathcal{M}, w \Vdash \varphi$ *we say that* φ *is* satisfied *in* $\mathcal{M}$ *at* w*. If* φ *is satisfied at all states in all models based on a frame* $\mathcal{F}$*, then we say that* φ *is valid on* $\mathcal{F}$ *and write* $\mathcal{F} \Vdash \varphi$*. If* φ *is valid on all frames, then we say that it is valid and write* $\Vdash \varphi$*.*

Now, you've certainly seen these definitions before — but if you want to understand contemporary modal logic you need to think about them in a certain way. Above all, please *don't* automatically think of models as a collection of "worlds" together with various "accessibility relations between worlds", and *don't* think of modalities as "non-classical logical symbols" suitable only for coping with intensional concepts such as necessity, possibility, and belief. Modal logic can be viewed in these terms, but it's a rather limited perspective. Instead, *think of models as relational structures*, or *multi-graphs*. That is, think of a model as an underlying set together with a collection of binary and unary relations. We use the modalities to talk about the binary relations, and the propositional symbols to talk about the unary relations.

Remark 1.4 (Kripke models are relational structures) *Let's make this precise. Consider a model* $\mathcal{M} = (W, \{R_\pi \mid \pi \in MOD\}, V)$*. The underlying frame* $(W, \{R_\pi \mid \pi \in MOD\})$ *is already presented in explicitly relational terms, and it is trivial to present the information in the valuation in same way: in fact* $\mathcal{M}$ *can be presented as the following relational structure* $\mathcal{M} = (W, \{R_\pi \mid \pi \in MOD\}, \{V(p) \mid p \in PROP\})$*.*

Why think in terms of relational structures? Two reasons. The first is: relational structures are ubiquitous. Virtually all standard mathematical structures can be viewed as relational structures, as can inheritance hierarchies, transition systems, parse trees, and other structures used in AI, computer science, and computational linguistics. Indeed, anytime you draw a diagram consisting of nodes, arcs, and labels, you have drawn some kind of relational structure. There are no preset limits to the applicability of modal logic: as it is a tool for talking about relational structures, it can be applied just about *anywhere*.

Secondly, relational structures are the models of *classical model theory* (see, for example, Hodges [35]). Thus there is nothing intrinsically "modal" about Kripke models, and we're certainly not forced to talk about them using modal languages. On the contrary, we can talk about models using *any* classical language we find useful (for example, a first-order, infinitary, fixpoint, or second-order language). Unsurprisingly, this means that modal and classical logic are systematically related.

Remark 1.5 (Modal logic is a fragment of classical logic) *To talk about a Kripke model in a classical language, all we have to do is view it as a relational structure (as described in the previous example) and then 'read off' from the signature (that is,*

MOD and PROP) the non-logical symbols we need, namely a MOD-indexed collection of two place relation symbols R_π*, and a PROP-indexed collection of unary relation symbols* P*,* Q*,* P'*,* Q'*, and so on. We then build formulas in the classical language of our choice.*

As modal languages and classical languages both talk about relational structures, it seems overwhelmingly likely that a systematic relationship exists between them. And in fact, the modal language (over PROP and MOD) can be translated into the best-known classical language of all, namely the first-order language (over PROP and MOD). Here are some clauses of the Standard Translation, a top-down translation which inductively maps modal to first-order formulas:

$$\begin{array}{lcl} ST_x(p) & = & \mathsf{P}(x),\ p \in PROP \\ ST_x(\neg\varphi) & = & \neg ST_x(\varphi) \\ ST_x(\varphi \wedge \psi) & = & ST_x(\varphi) \wedge ST_x(\psi) \\ ST_x(\langle\pi\rangle\varphi) & = & \exists y(x\mathsf{R}_\pi y \wedge ST_y(\varphi)) \\ ST_x([\pi]\varphi) & = & \forall y(x\mathsf{R}_\pi y \rightarrow ST_y(\varphi)). \end{array}$$

Here x *is a fixed but arbitrary free variable. In the fourth and fifth clause, the variable* y *can be any variable not used so far in the translation. The clauses governing* ST_y *are analogous to those given for* ST_x*; in particular, the clauses for the modalities introduce a new variable (say* z*) and so on. For any modal formula* φ*,* $ST_x(\varphi)$ *is a first-order formula containing exactly one free variable (namely* x*), and it is easy to see that* $\mathcal{M}, w \Vdash \varphi$ *iff* $\mathcal{M} \models ST_x(\varphi)[w]$ *(where* $\models$ *denotes the first-order satisfaction relation and* $[w]$ *means assign the state* w *to the free variable* x *in* $ST_x(\varphi)$*). The equivalence can be proved by induction, but it should be self-evident: the Standard Translation is simply a reformulation of the clauses of the Kripke satisfaction definition.*

There are also non-trivial links between modal logic and infinitary logic, fixed-point logic, and second-order logic; in particular, modal validity is intrinsically second-order. For further discussion, see Blackburn, de Rijke, and Venema [14].

In short, modal logic is not some mysterious non-classical intensional logic, and modalities are not strange new devices. On the contrary, *modalities are simply macros that handle quantification over accessible states.*

This, of course, leads to another question. OK — so we *can* use modal logic when working with relational structures — but why bother if it's really just a disguised way of doing classical logic? I think the following two answers are the most important: modal logic brings *simplicity* and *perspective*.

Simplicity comes in a variety of forms. For a start, modal representations are often clean and compact: modalities pack a useful punch into a readable notation. Moreover, modal logic often brings us back to the realms of the computable: while the first-order logic over MOD and PROP is *undecidable* (whenever MOD is non-empty), its modal logic is *decidable* (in fact, PSPACE-complete).

Perspective is more subtle. Modal languages talk about relational structures in a special way: they take an *internal* and *local* perspective on relational structure. When we evaluate a modal formula, we place it *inside* the model, at some particular state w (the *current state*). The satisfaction clause (and in particular, the clause for the modalities) allow us to scan other states for information — but we're only allowed to scan states reachable from the current state. The reader should think of a modal formula as a little automaton, placed at some point on a graph, whose task is to explore

the graph by visiting accessible states. This internal, local, perspective is responsible for many of the attractive mathematical properties of modal logic. Moreover, it makes modal representations ideal for many applications. Here's a classic example:

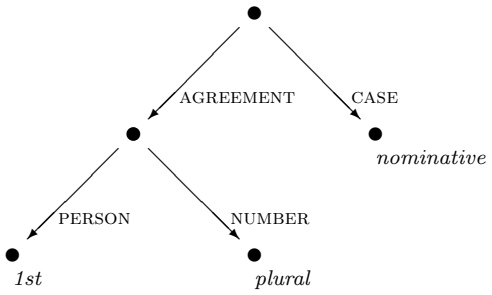
Example 1.6 (Temporal logic) *We'll be seeing a lot of the bimodal language with* $MOD = \{\mathsf{F}, \mathsf{P}\}$ *in this paper: the modality* $\langle \mathsf{F} \rangle$ *means "at some Future state", and* $\langle \mathsf{P} \rangle$ *means "at some Past state". To reflect this temporal interpretation, we usually interpret this language on frames of the form* $(T, <)$ *that can plausibly be thought of as 'flows of time'. For example, if we think of time as a branching structure,* $(T, <)$ *might be some kind of tree, and if we want a linear view of time,* $(T, <)$ *might be* $(\mathbb{Z}, <)$ *(the integers in their usual order). When interpreting the language on such frames we insist that* R_{F} *is* $<$*, and* R_{P} *is its converse; that is, as required, we ensure that* $\langle \mathsf{F} \rangle$ *looks forward along the flow of time, and* $\langle \mathsf{P} \rangle$ *backwards.*

Consider the formula $\langle \mathsf{P} \rangle$Mia-unconscious*. This is true iff we can look back in time from the current state and see a state where Mia is unconscious. Similarly* $\langle \mathsf{F} \rangle$Mia-unconscious *requires us to scan the states that lie in the future looking for one where Mia is unconscious. Thus these two formulas work similarly to the English sentences* Mia has been unconscious *and* Mia will be unconscious*: these sentences don't specify an absolute time for Mia's unconsciousness (which we could do by giving a date and time), rather they locate it relative to the time of utterance. In short, English and other natural languages exploit the fact that human beings live in time, and modal logic models this neatly.*

This situated perspective can be lifted to more interesting temporal geometries. For example, we could regard temporal states as unbroken intervals of time, add new modalities such as $\langle \mathsf{SUB} \rangle$ *(meaning "at some SUBinterval of the current state") and* $\langle \mathsf{SUP} \rangle$ *(meaning "at some SUPerinterval of the current state"). Then a formula of the form* $\langle \mathsf{SUB} \rangle \langle \mathsf{F} \rangle \langle \mathsf{SUP} \rangle p$ *means "by looking down to a subinterval, and then forward to the future, and then up to a superinterval, it is possible to find a state where* p *is true". Halpern and Shoham [34] take this idea to its ultimate conclusion: abstracting from the work of James Allen [1], they present a modal logic which allows all possible relationships between two closed intervals over a linear flow of time to be explored 'from the inside'.*

Nowadays, few modal logicians regard modal logic as a non-classical logic, and they certainly don't feel tied to any of the traditional interpretations of modal machinery. On the contrary, since the early 1970s modal logic has been explored as a subsystem of various classical logics, and it is now clear that modal logic are a very special part of classical logic. Indeed, modal languages are in many respects so natural, that — as modal logicians love to point out — it's not particularly surprising that they have been independently reinvented by other research communities that make use of relational structures. Let's look at two well known examples.

Example 1.7 (Feature logic) *Feature structures are widely used in unification-based approaches to natural language. In essence, feature structures are multigraphs that represent linguistic information:*



Computational linguists have a neat notation for talking about feature structures: Attribute-Value Matrices (AVMs). Here's an example:

$$\begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & \text{1st} \\ \text{NUMBER} & \text{plural} \end{bmatrix} \\ \text{CASE} & -\text{dative} \end{bmatrix}$$

This AVM is a partial description in the above feature structure — it's satisfied in that structure at the root node. The AVM describes a feature structure in which the AGREEMENT *transition leads to a node from which* PERSON *and* NUMBER *transitions lead to the information 1st and plural respectively, and if you work down the left hand side of the previous diagram from the root you'll find this structure. The AVM also demands a* CASE *transition from the root node that does not lead to the information dative. The feature structure depicted above also satisfies this requirement, for the* CASE *transition leads to a node bearing the information nominative.*

Now, this all sounds very modal — and indeed, the AVM is a notational variant of the following formula:

$$\begin{array}{ll} & \langle \text{AGREEMENT} \rangle (\langle \text{PERSON} \rangle \mathit{1st} \wedge \langle \text{NUMBER} \rangle \mathit{plural}) \\ \wedge & \langle \text{CASE} \rangle \neg \mathit{dative} \end{array}$$

Example 1.8 (Description logic/Terminological logic) *In description logic, concept languages are used to build knowledge bases. An important part of the knowledge base is called the TBox (or terminology). This is a collection of concept macros defined over the primitive concept names using booleans and role names. For example, the concept of being a hired killer for the mob is true of any individual who is a killer and employed by a gangster, and we can define this in the description language $\mathcal{ALC}$ using the following expression:*

$$\mathsf{killer} \sqcap \exists \text{EMPLOYER}.\mathsf{gangster}$$

Here killer *and* gangster *are concept names,* EMPLOYER *is a role name, and $\sqcap$ is a boolean (intersection). This expression means exactly the same thing as the following modal formula:*

$$\mathsf{killer} \wedge \langle \text{EMPLOYER} \rangle \mathsf{gangster}$$

*Indeed, as Schild [49] pointed out, any $\mathcal{ALC}$ expression corresponds to a modal formula: simply replace occurrences of $\sqcap$ by $\wedge$, $\sqcup$ by $\vee$, $\exists \mathrm{R}$ by $\langle \mathrm{R} \rangle$, and $\forall \mathrm{R}$ by $[\mathrm{R}]$ (both formalisms typically use the symbol $\neg$ to denote boolean-complement/negation, so occurrences of $\neg$ can be left in place). This correspondence lifts to many stronger concept languages: number restrictions correspond to counting modalities, mutually converse roles correspond to mutually converse modalities, and commonly used role constructors (for example, for forming the transitive closure of a role) correspond to the modality constructors of Propositional Dynamic Logic (*PDL*).*

Summing up, modal logic is a well-behaved and intuitively natural fragment of classical logic. Over the past 25 years, modal logicians have explored and extended this fragment in many ways. By introducing modal operators of arbitrary arities, they have made it possible to work with relational structures containing relations of any arity. By evaluating formulas at *sequences* of states (as is done in *multidimensional modal logic*; see Marx and Venema [39]) they have generalized the notion of perspective. By introducing *logical modalities* (see Goranko and Passy [33] and de Rijke [47]) they have shown how to introduce certain forms of *globality* into modal logic while retaining (and in certain respects improving) their desirable properties. Indeed, in recent work on the *guarded fragment* (see Andréka, van Benthem, and Németi [2]) they have shown that it is even possible to "export" the locality intuition back to classical logic; this line of work has unearthed several previously unknown decidable fragments of first-order (and other) classical logics. For a detailed account of contemporary modal logic, see Blackburn, De Rijke, and Venema [14].

So modal logicians have a lot to be proud of. But for all these achievements, something is missing. What exactly?

2 The Trouble with Modal Logic

Carlos Areces summed it up neatly: there is an *asymmetry* at the heart of modal logic. Although states are crucial to Kripke semantics, nothing in modal syntax get to grips with them. This leads to (at least) two kinds of problem. For a start, it means that for many applications modal logic is *not* an adequate representation formalism. Moreover, it makes it difficult to devise usable modal *reasoning* systems.

Example 2.1 (Temporal logic) *Although the temporal language with modalities $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$ neatly captures the perspectival nature of natural language tenses, it fails to get to grips with a linguistic fact of equal importance: many tenses are referential. An utterance of* Vincent accidentally squeezed the trigger *doesn't mean that at some completely unspecified past time Vincent did in fact accidentally squeeze the trigger, it means that at some* particular*, contextually determined, past time he did so. The natural representation, $\langle \mathsf{P} \rangle$*Vincent-accidentally-squeeze-the-trigger*, fails to capture this.*

Similarly, while it's certainly possible to abstract elegant modal logics from the work of James Allen, such abstractions amputate a central feature of his work: reference to specific intervals. Allen's formalism includes the notation **Hold**(P, i) *meaning "the property* P *holds at the interval i", and* **Hold** *plays a key role in his approach to temporal knowledge representation. This construction is not present in the modal logic of Halpern and Shoham.*

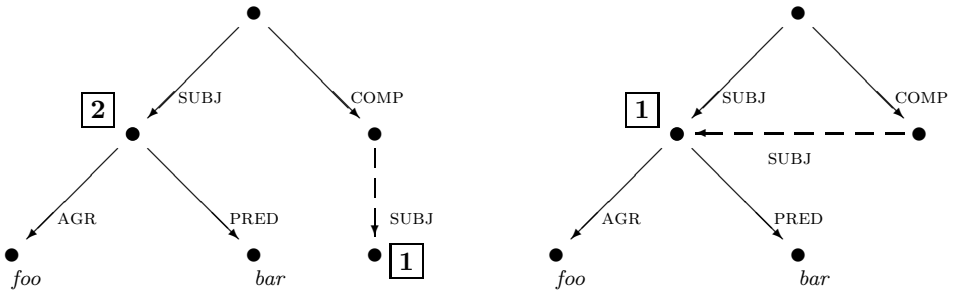
And there are deeper limitations, centered on the notion of validity. Suppose we

are working with the temporal language in $\langle \mathsf{F} \rangle$ *and* $\langle \mathsf{P} \rangle$*. Can we write down a formula that is valid on every transitive frame, and not valid on any others? That is, can we define transitivity? Sure:* $\langle \mathsf{F} \rangle \langle \mathsf{F} \rangle p \rightarrow \langle \mathsf{F} \rangle p$ *does so. OK: but can we write down a formula valid on precisely the asymmetric frames (that is, frames* $(T, <)$ *such that* $\forall xy(x < y \rightarrow y \not< x)$*)? Try what you like, you won't find any such formula: a central property of flows of time is invisible to modal representations.*

Example 2.2 (Feature Logic) *While AVM notation is related to modal logic, it offers something new: it lets us name specific nodes in feature structures. Consider the following AVM:*

$$\begin{bmatrix} \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{AGR} & \text{foo} \\ \text{PRED} & \text{bar} \end{bmatrix} \\ \text{COMP} & [\text{SUBJ} \quad \boxed{1}\,] \end{bmatrix}$$

The 'tag $\boxed{1}$ *names a point in the feature structure. This AVM demands that the node we reach by following the* SUBJ *transition from the root is also the node we reach by first taking a* COMP *transition from the root and then taking a* SUBJ *transition. No matter which path we take, we have to end up at the node tagged* $\boxed{1}$*. For this reason, the AVM is not satisfied on the left hand feature below (which otherwise gets everything right) but is satisfied by the right hand feature structure:*



Thus AVM notation is *not* a notational variant of ordinary multimodal logic: it's strictly stronger. So are many description logics:

Example 2.3 (Description logic) *Description logic lets us reason about specific individuals — in fact, it lets us do so in two distinct ways. First, knowledge bases need not consist of just a TBox — they can also contain an ABox. In the ABox (or assertional component) we specify how properties and roles apply to specific individuals. For example, to assert that Vincent is a gunman we add* $\mathsf{Vincent}{:}\mathsf{gunman}$ *to the ABox, and to insist that Pumpkin loves Honey-Bunny we add (*$\mathsf{Pumpkin}, \mathsf{Honey\text{-}Bunny}$*):*LOVES*.*

Now, the assertional level is a separate level in the knowledge base, so such specifications aren't written in the underlying concept language (in essence they're statements in a constraint language that manipulates formulas of the concept language). But some description languages push matters further: just as feature logic does, they allow reference to individuals to be integrated into the underlying representation formalism itself, thus allowing assertions about individuals to be integrated into the TBox. This is done via the one-of operator $\mathcal{O}$*. The notation* $\mathcal{O}(\mathsf{Jules}, \ldots, \mathsf{Vincent})$ *picks out one*

of the individuals Jules, ... , Vincent, *and* $\mathcal{O}$(Mia) *picks out Mia. In short, we now have a concept language rich enough to refer to specific individuals. Such a concept language is not a notational variant of ordinary multimodal logic, or even multimodal logic enriched with (say) counting modalities and* PDL*-like constructs: it offers a novel form of expressivity.*

There is a method (introduced in the late 1960s by Arthur Prior) which allows reference to states to be incorporated into modal logic. But although Prior's idea has attracted a handful of advocates (see the Guide to the Literature at the end of the paper) it's never been part of the modal mainstream. On the other hand, description logicians such as De Giacomo [22] have realized its relevance. This method — hybridization — is central to the paper, and I'll introduce it shortly.

In short, the asymmetry underlying orthodox modal logic translates into obvious weaknesses as a representation formalism. The same asymmetry leads to problems with *reasoning*. Until recently, modal proof theory was a relatively neglected topic. Traditionally, modal logicians have been content to formulate modal proof systems as Hilbert-style axiomatizations, this being enough to get on with the topics that interested them with a minimum of syntactic fuss. But it resulted in few *usable* modal proof systems available, and little in the way of general proof-theoretical results.

An important exception to this was Fitting's [25] groundbreaking work on *prefixed tableau systems*. Fitting's work can be viewed as a precursor to Gabbay's [26] work on *labelled deduction*. In essence, Gabbay's proposal is to develop a metalinguistic algebra of labels that can act as the motor for modal deduction. Another recent general approach, *display calculus* (see Kracht [37]), though very different from labelled deduction, also makes use of novel metalinguistic machinery. Display calculus is an extension of sequent calculus which introduces additional notation to allow us to freely manipulate object language formulas (in much the same way as a school child rewrites polynomial equations).

Now, first-order proof theory does not require this kind of metalinguistic support. This is because first-order languages are expressive enough to support the key deduction steps at the *object* level. If we find a representation formalism that is *not* capable of doing this, but needs to be augmented by a rich metatheoretic machinery, this is a signal that something is missing. Modal logic seems to be such a formalism — what exactly does it lack?

If we look at the Fitting-Gabbay tradition, an answer practically leaps off the page: we need to be able to deal with states *explicitly*. We need to be able to name them, reason about their identity, and reason about the transitions that are possible between them. In essence, labelled deduction in its various forms supplies metalinguistic equipment for carrying out these tasks, and this leads to modally natural proof systems. In particular, labelled deduction successfully captures the key intuition underlying Kripke semantics, that of a little automaton working it's way through a graphlike structure — except that the automaton's *deductive* task is to try and *build* such a structure, not explore a pre-existing one.

Summing up, whether we think about representation or reasoning the conclusion is the same: modal logic's lack of mechanisms for dealing with states explicitly is a genuine weakness.

3 Hybrid Logic

Hybrid languages provide a truly *modal* solution to this problem. Modal logic may not be perfect — but it's certainly a most remarkable fragment of classical logic. How can we add reference to states without destroying it?

Let's go back to basics. Modal logic allows us to form complex formulas out of atomic formulas using booleans and modalities. There's only formulas, nothing else. So if we want to name states and remain modal, we should find a way of naming states using formulas. We can do this by introducing a second sort of atomic formula: *nominals*. Syntactically these will be ordinary atomic formulas, but they will have an important *semantic* property: nominals will be true at exactly *one* point in any model; nominals 'name' this point by being true there and nowhere else. Let's make this idea precise — and improve it in one respect, by adding *satisfaction operators*.

Definition 3.1 (Hybrid multimodal languages) *Let NOM be a nonempty set disjoint from PROP and MOD. The elements of NOM are called nominals, and we typically write them as i, j, k and l. We define the hybrid multimodal language (over PROP, NOM, and MOD) to be the:*

$$WFF := i \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \langle\pi\rangle\varphi \mid [\pi]\varphi \mid @_i\varphi.$$

For any nominal i, we shall call the symbol sequence $@_i$ a satisfaction operator.

Remark 3.2 (Nominals and satisfaction operators) *As promised, nominals are formulas. What are satisfaction operators? In essence, a simple way of further exploiting the presence of nominals: $@_i\varphi$ means "go to the point named by i (that is, the unique point where i is true) and see if φ is true there". That is, $@_i\varphi$ is a way of asserting — in the object language — that φ is satisfied at a particular point. Formulas of the form $@_i\varphi$ and $\neg@_i\varphi$ are called satisfaction statements.*

Definition 3.3 (Hybrid models, satisfaction, and validity) *A hybrid model is a triple $(W, \{R_\pi \mid \pi \in MOD\}, V)$ where $(W, \{R_\pi \mid \pi \in MOD\})$ is a frame and V is a* hybrid valuation. *A hybrid valuation is a function with domain PROP$\cup$NOM and range $Pow(W)$ such that for all nominals i, $V(i)$ is a singleton subset of W. We call the unique state in $V(i)$ the denotation of i. We interpret hybrid languages on hybrid models by adding the following two clauses to the Kripke satisfaction definition:*

$$\begin{array}{lll} \mathcal{M}, w \Vdash i & \textit{iff} & w \in V(i), \textit{ where } i \in \mathrm{NOM} \\ \mathcal{M}, w \Vdash @_i\varphi & \textit{iff} & \mathcal{M}, w' \Vdash \varphi, \textit{ where } w' \textit{ is the denotation of } i. \end{array}$$

If φ is satisfied at all states in all hybrid models based on a frame $\mathcal{F}$, then we say that φ is valid on $\mathcal{F}$ and write $\mathcal{F} \Vdash \varphi$. If φ is valid on all frames, then we say that it is valid and write $\Vdash \varphi$.

Remark 3.4 (Hybrid logic is modal) *Hybrid languages contain only familiar modal mechanisms: nominals are atomic formulas, and satisfaction operators are actually normal modal operators (that is: for any nominal i, $@_i(\varphi \rightarrow \psi) \rightarrow (@_i\varphi \rightarrow @_i\psi)$ is valid; and if φ is valid, then so is $@_i\varphi$).*

Moreover, like multimodal logic, hybrid logic is a fragment of classical logic: indeed, it is easy to extend the Standard Translation to hybrid logic. Divide the first-order

variables into two sets such that one contains the reserved variable x and the variables used to translate familiar modalities, while the other contains a first-order variable x_i for every nominal i. Define:

$$\begin{array}{lcl} ST_x(i) & = & x = x_i,\ i \in NOM \\ ST_x(@_i\varphi) & = & (ST_x(\varphi))[x_i/x] \end{array}$$

Clearly $\mathcal{M}, w \Vdash \varphi$ iff $\mathcal{M} \models ST_x(\varphi)[w, V(i), \ldots, V(j)]$, where x, x_i, $\ldots$ x_j are the free variables in $ST_x(\varphi)$. Nominals correspond to free variables, and (as the substitution $[x_i/x]$ makes clear) satisfaction operators let us switch our perspective from the current state to named states.

So far, so modal — but what about computational complexity? No change. As Areces, Blackburn and Marx [4] show, hybrid logic is (up to a polynomial) no more complex than multimodal logic: deciding the validity of hybrid formulas is a PSPACE-complete problem.

Remark 3.5 (Hybrid logic is hybrid) *Any modal logic is a fragment of classical logic — but hybrid logic takes matters a lot further. The near-atomic satisfaction statement $@_i j$ asserts that the states named by i and j are identical, thus we have incorporated part of the classical theory of equality. Similarly $@_i\langle\pi\rangle j$ means that the state named by j is an R_π-successor of the state named by i, so we've incorporated the classical ability to make assertions about the relations that hold between specific states. Thus hybrid logic is a genuine hybrid: it brings to modal logic the classical concepts of identity and reference.*

With this extra classical power at out disposal, it is straightforward to fix the representational problems noted in the previous section.

Example 3.6 (Temporal logic) *First, although* Vincent accidentally squeezed the trigger *can't be correctly represented in the ordinary temporal language in $\langle \mathsf{F}\rangle$ and $\langle \mathsf{P}\rangle$, it can be with the help of nominals: $\langle \mathsf{P}\rangle(i \wedge$*Vincent-accidentally-squeeze-the-trigger*$)$ locates the trigger-squeezing not merely in the past, but at a specific temporal state there: the one named by i.*

Second, if we want to work with interval-based temporal models, we can now do so in a way that is faithful to the work of James Allen: the satisfaction statement $@_i\varphi$ is a clear analog of Allen's **Hold**(i, φ) *construct. More on this in Section 6.*

Third, we also solve the deeper issue concerning definability: $i \rightarrow \neg\langle \mathsf{F}\rangle\langle \mathsf{F}\rangle i$ defines asymmetry (that is, it is valid on all asymmetric frames and no others). More on this in Section 5.

Example 3.7 (Feature logic) *Nominals correspond to tags. Consider once more the problematic AVM:*

$$\left[\begin{array}{lll} \textsc{subj} & \boxed{1} & \left[\begin{array}{ll} \textsc{agr} & \textsc{foo} \\ \textsc{pred} & \textsc{bar} \end{array}\right] \\ \textsc{comp} & [\textsc{subj} & \boxed{1}\,] \end{array}\right]$$

This corresponds to the following L^N wff:

$$\begin{array}{ll} & \langle\textsc{subj}\rangle(i \wedge \langle\textsc{agr}\rangle\textsc{foo} \wedge \langle\textsc{pred}\rangle\textsc{bar}) \\ \wedge & \langle\textsc{comp}\rangle\langle\textsc{subj}\rangle i \end{array}$$

And in fact, AVM notation is essentially a two-dimensional notation for multimodal logic with nominals. For more on feature logic as hybrid logic, see Blackburn [10], Blackburn and Spaan [17], and Reape [45, 46] (and see Bird and Blackburn [9] for related ideas in phonology).

Example 3.8 (Description Logic) *The TBoxes of the concept language $\mathcal{ALCO}$ (that is, $\mathcal{ALC}$ enriched with the $\mathcal{O}$ operator mentioned in Example 2.3) is a notational variant of the @-free fragment of hybrid multimodal logic. First, every nominal corresponds to an expression of the form $\mathcal{O}(i)$. Conversely, every $\mathcal{ALCO}$ expression of the form $\mathcal{O}(i, \ldots, j)$ corresponds to the formula $i \vee \cdots \vee j$.*

Furthermore, @ has a natural description logic interpretation. The ABox specification $i{:}\varphi$ corresponds to the satisfaction statement $@_i\varphi$, and the specification $(i, j) : \mathrm{R}$ corresponds to $@_i\langle \mathrm{R}\rangle j$. But whereas ABox specifications are constraints stated at a separate representational level, their hybrid equivalents are part of the object language. In effect, hybrid multimodal logic is an extension of $\mathcal{ALCO}$ which fully integrates ABox specifications into the concept language (without moving us out of PSPACE). For more on description logic as hybrid logic, see De Giacomo [22], Blackburn and Tzakova [19], Areces and de Rijke [6], and (in spite of its title) Areces, Blackburn and Marx [3].

4 Hybrid Reasoning

Nominals and @ make it possible to create names for states, and to reason about state identity and the way states are linked. This give us enough classical power in the object language to capture the modal locality intuition (recall the little automaton exploring/building graphs) *without* requiring elaborate metatheoretic proof machinery. Hybrid deduction is a form of labelled deduction — but it's labelled deduction that has been internalized into the object language. I'll formulate hybrid reasoning as an unsigned tableau system. We'll need two groups of rules. Here's the first:

$$\frac{@_s\neg\varphi}{\neg@_s\varphi}\ [\neg] \qquad \frac{\neg@_s\neg\varphi}{@_s\varphi}\ [\neg\neg]$$

$$\frac{@_s(\varphi\wedge\psi)}{\begin{array}{c}@_s\varphi\\ @_s\psi\end{array}}\ [\wedge] \qquad \frac{\neg@_s(\varphi\wedge\psi)}{\neg@_s\varphi \mid \neg@_s\psi}\ [\neg\wedge]$$

$$\frac{@_s@_t\varphi}{@_t\varphi}\ [@] \qquad \frac{\neg@_s@_t\varphi}{\neg@_t\varphi}\ [\neg@]$$

$$\frac{@_s\langle\pi\rangle\varphi}{\begin{array}{c}@_s\langle\pi\rangle a\\ @_a\varphi\end{array}}\ [\langle\pi\rangle] \qquad \frac{\neg@_s\langle\pi\rangle\varphi \quad @_s\langle\pi\rangle t}{\neg@_t\varphi}\ [\neg\langle\pi\rangle]$$

$$\frac{@_s[\pi]\varphi \quad @_s\langle\pi\rangle t}{@_t\varphi}\ [[\pi]] \qquad \frac{\neg@_s[\pi]\varphi}{\begin{array}{c}@_s\langle\pi\rangle a\\ \neg@_a\varphi\end{array}}\ [\neg[\pi]]$$

In these rules, s and t are metavariables over nominals, and a is a metavariable over new nominals (that is, nominals not used so far in the tableau construction). The

rules for $\vee$ and $\rightarrow$ are obvious variants of the rules for $\wedge$ (we'll see both rules when we give some examples).

Remark 4.1 (The first group internalizes the satisfaction definition) *These rules use the resources available in hybrid logic to mimic the Kripke satisfaction definition: they draw conclusions from the input to each rule (the formula(s) above the horizontal line) to the output (the formula(s) below the line). For example, the $\wedge$-rule says that if $\varphi \wedge \psi$ is true at s, then both φ and ψ are true at s, while it's dual rule $\neg\wedge$ (a branching rule) says that if $\varphi \wedge \psi$ is false at s, then either φ or ψ is false at s. Note that both the $[\pi]$-rule and the $\neg\langle\pi\rangle$-rule take two input formulas, one of which (the minor premiss) is a formula of the form $@_s\langle\pi\rangle t$. For example, the $[\pi]$-rule says that if a pair of formulas of the form $@_s[\pi]\varphi$ and $@_s\langle\pi\rangle t$ can be found on some branch of the tableau, we are free to extend that branch by adding $@_t\varphi$ — a clear reflection of the Kripke semantics for $[\pi]$. Already first-order ideas are creeping into the system: this rule trades on the fact that hybrid logic is strong enough to make statements about state succession (using near-atomic satisfaction statements of the form $@_s\langle\pi\rangle t$).*

But it is with the $\langle\pi\rangle$- and $[\pi]$-rules that first-order ideas really make themselves felt. What do we know when a formula of the form $\langle\pi\rangle\varphi$ is true at s? The Kripke satisfaction definition gives us the answer: we know that (1) we can make an R_π transition from s to some state, and (2) at this R_π-successor state, φ is true. The $\langle\pi\rangle$-rule captures this idea: it tells us to (1) introduce a new nominal a to name the successor state, and (2) insist that φ is true at a. Recall that in first-order reasoning, existential quantifiers are eliminated by introducing new parameters. In effect, the $\langle\pi\rangle$-rule uses nominals to exploit this first-order idea. Incidentally: we don't apply the $\langle\pi\rangle$-rule to formulas of the form $@_s\langle\pi\rangle\varphi$ where φ is a nominal. Doing so is pointless, for it would simply create a new name for a state that already had a name.

But we need a second group of rules. Nominals and @ come with a certain amount of logic built in: they provide theories of state equality and state succession. Just as we need to add special rules or axioms to first-order logic to handle the equality symbol correctly, we need additional mechanisms for nominals and @:

$$\frac{[s \text{ on branch}]}{@_s s}\ \text{[Ref]} \qquad \frac{@_t s}{@_s t}\ \text{[Sym]} \qquad \frac{@_s t \quad @_t\varphi}{@_s\varphi}\ \text{[Nom]} \qquad \frac{@_s\langle\pi\rangle t \quad @_t t'}{@_s\langle\pi\rangle t'}\ \text{[Bridge]}$$

Remark 4.2 (The second group is essentially a classical rewrite system) *The* Ref *rule says that if a nominal s occurs in any formula on a branch, then we are free to add $@_s s$ to that branch; this is clearly an analog of the first-order reflexivity rule for $=$, just as the* Sym *rule is an analog of the first-order symmetry rule for $=$. What about transitivity? From $@_s t$ and $@_t t'$ we should be able to conclude $@_s t'$. But this is a special case of* Nom*, namely when φ is chosen to be a nominal t'. More generally, Nom ensures that identical states carry identical information, while* Bridge *ensures that states are coherently linked. In first-order terms, these rules ensure that state identity is not merely an equivalence relation but a congruence.*

As with any tableau system, we prove formulas by systematically trying to falsify them. Suppose we want to prove φ. We choose a nominal (say i) that does not occur in φ (this acts as a name for the falsifying state that is supposed to exist),

prefix φ with $\neg @_i$, and start applying rules. If the tableau closes (that is, if every branch contains some formula and its negation), then φ is proved. On the other hand, suppose we reach a stage where we have applied the appropriate connective rule to every complex formula (or in the case of $[\pi]$-formulas, we have applied the $[\pi]$-rule to every pair of formulas of the form $@_s[\pi]\varphi$, $@_s\langle\pi\rangle t$ on the same branch; and analogously for $\neg\langle\pi\rangle$-formulas) and no application of the rewrite rules yields anything new. If the tableau we have constructed contains open branches (that is, branches not containing conflicting formulas), then φ is not valid (and hence not provable), and the near-atomic satisfaction statements on the open branch specify a countermodel.

Example 4.3 (A standard multimodal validity) *Let's start with an example from ordinary multimodal logic:* $\langle\pi\rangle(p \vee q) \rightarrow \langle\pi\rangle p \vee \langle\pi\rangle q$ *is valid (for any modality* $\langle\pi\rangle$*), hence this formula should be provable. Here's how to do it:*

1		$\neg @_i(\langle\pi\rangle(p \vee q) \rightarrow \langle\pi\rangle p \vee \langle\pi\rangle q)$	
2		$@_i\langle\pi\rangle(p \vee q)$	$1, \neg\rightarrow$
$2'$		$\neg @_i(\langle\pi\rangle p \vee \langle\pi\rangle q)$	*Ditto*
3		$\neg @_i\langle\pi\rangle p$	$2', \neg\vee$
$3'$		$\neg @_i\langle\pi\rangle q$	*Ditto*
4		$@_i\langle\pi\rangle j$	$2, \langle\pi\rangle$
$4'$		$@_j(p \vee q)$	*Ditto*
5		$\neg @_j p$	$3, 4, \neg\langle\pi\rangle$
6		$\neg @_j q$	$3', 4, \neg\langle\pi\rangle$
7	$@_j p$	| $@_j q$	$4', \vee$
	✠ 5, 7 ✠	✠ 6, 7 ✠	

In short, we start with one initial state (namely i*) and then use the tableau rules to reason about what must hold there. At line 4 we use the* $\langle\pi\rangle$*-rule to introduce a new state name, namely* j*. We continue to reason about the way information must be distributed across these two states until we are forced to conclude that there is no coherent way of doing so.*

Example 4.4 (A genuinely hybrid validity) *The previous example gives only the barest hint of what the system can do. Here's a more interesting example, which shows that hybrid reasoning not merely makes use of nominals and @, but also gets to grip with the logic of state identity and succession they embody.*

Suppose we're working with a language with three modalities. To emphasize the geometric intuitions underlying hybrid reasoning, let's call these $\langle\text{VERT}\rangle$*,* $\langle\text{HOR}\rangle$ *and* $\langle\text{DIAG}\rangle$ *(for vertical, horizontal and diagonal) respectively. Now,* $\langle\text{HOR}\rangle\langle\text{VERT}\rangle(i \wedge p) \wedge \langle\text{DIAG}\rangle i \rightarrow \langle\text{DIAG}\rangle p$ *is valid (for there's only one state named* i*) and we can prove*

it as follows:

1	$\neg @_j(\langle \text{HOR} \rangle \langle \text{VERT} \rangle (i \wedge p) \wedge \langle \text{DIAG} \rangle i \rightarrow \langle \text{DIAG} \rangle p)$	
2	$@_j(\langle \text{HOR} \rangle \langle \text{VERT} \rangle (i \wedge p) \wedge \langle \text{DIAG} \rangle i)$	$1, \neg\rightarrow$
$2'$	$\neg @_j \langle \text{DIAG} \rangle p$	*Ditto*
3	$@_j \langle \text{HOR} \rangle \langle \text{VERT} \rangle (i \wedge p)$	$2, \wedge$
$3'$	$@_j \langle \text{DIAG} \rangle i$	*Ditto*
4	$@_j \langle \text{HOR} \rangle k$	$3, \langle \text{HOR} \rangle$
$4'$	$@_k \langle \text{VERT} \rangle (i \wedge p)$	*Ditto*
5	$@_k \langle \text{VERT} \rangle l$	$4', \langle \text{VERT} \rangle$
$5'$	$@_l (i \wedge p)$	*Ditto*
6	$@_l i$	$5', \wedge$
$6'$	$@_l p$	*Ditto*
7	$@_i l$	$6, Sym$
8	$@_i p$	$6', 7, Nom$
9	$\neg @_i p$	$2', 3', \neg\langle \text{DIAG} \rangle$
	✠ 8, 9 ✠	

Think in terms of a graph-building automaton: it creates an initial state named i, generates successor states j, k and l, and reasons about the way information must be distributed over them until it becomes clear that there is no way to construct a countermodel.

Remark 4.5 (There are other approaches) *I have presented hybrid reasoning as an unsigned tableau system, but we are not forced to do this, and the underlying graph construction intuition come through in a range of proof styles. For example, Seligman [52] presents sequent and natural deduction systems with much the same geometrical flavor (indeed Seligman motivates his rules by discussing what a logic of spatial locations should look like). The same is true of Tzakova's [55] Fitting-style indexed tableau approach, Demri's [23] sequent system for the $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$ language enriched with nominals but without @, and Konikowska's [36] sequent based approach to the logic of relative similarity.*

One last point. The link with orthodox modal labelled deduction should now be clear — but there is also a link with description logic: hybrid reasoning is a form of ABox reasoning. The tableau system manipulates satisfaction statements, which are essentially ABox specifications (recall Example 3.8).

5 Other Frame Classes

The tableau system is (sound and) complete in the following sense. Let us say that a formula φ is *tableau provable* iff there is a closed tableau with $\neg @_i \varphi$ as its root (where i is a nominal not occurring in φ). Then:

Theorem 5.1 *φ is tableau provable iff φ is valid.*

Proof. Soundness is straightforward. A completeness proof for unimodal languages is given in Blackburn [13] using a Hintikka set argument; it extends straightforwardly to multimodal languages. ∎

So far so good — but valid means "true in all states in any hybrid model based on *any* frame", and often we only care about models based on frames with certain properties, and we want to reason in the stronger logics such frames give rise to.

In many cases hybrid reasoning adapts straightforwardly to cope with such demands. In particular, if we use *pure* formulas (that is, formulas containing no propositional variables) there is a straightforward link between *defining* a class of frames and *reasoning* about the frames in that class. A formula φ *defines* a class of frames F iff φ is valid on all the frames in F and falsifiable on any frame not in F. A formula defines a property of frames (such as transitivity) iff it defines the class of frames with that property. So: what can pure formulas define?

Example 5.2 (Pure formulas and frame definability) *Consider the temporal language in $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$. Using pure formulas, we can define a number of properties relevant to temporal logic:*

$@_i \neg \langle \mathsf{F} \rangle i$	$\forall x \neg (xR_{\mathsf{F}}x)$ *(Irreflexivity)*
$@_i \neg \langle \mathsf{F} \rangle \langle \mathsf{F} \rangle i$	$\forall xy(xR_{\mathsf{F}}y \rightarrow \neg yR_{\mathsf{F}}x)$ *(Asymmetry)*
$@_i[\mathsf{F}](\langle \mathsf{F} \rangle i \rightarrow i)$	$\forall xy(xR_{\mathsf{F}}y \wedge yR_{\mathsf{F}}x \rightarrow x = y)$ *(Antisymmetry)*
$\langle \mathsf{F} \rangle \langle \mathsf{F} \rangle i \rightarrow \langle \mathsf{F} \rangle i$	$\forall xyz(xR_{\mathsf{F}}y \wedge yR_{\mathsf{F}}z \rightarrow xR_{\mathsf{F}}z)$ *(Transitivity)*
$\langle \mathsf{F} \rangle i \rightarrow \langle \mathsf{F} \rangle \langle \mathsf{F} \rangle i$	$\forall xy(xR_{\mathsf{F}}y \rightarrow \exists z(xR_{\mathsf{F}}z \wedge zR_{\mathsf{F}}y))$ *(Density)*
$@_i \langle \mathsf{F} \rangle j \vee @_i j \vee @_j \langle \mathsf{F} \rangle i$	$\forall xy(xR_{\mathsf{F}}y \vee x = y \vee yR_{\mathsf{F}}z)$ *(Trichotomy)*

The properties just listed only tell us about R_{F} — but a far more basic property of frames is needed for temporal logic, namely that R_{F} and R_{P} be mutually converse relations. This can also be defined using pure formulas. First note that the following relations between R_{F} and R_{P} are definable:

$@_i[\mathsf{F}]\langle \mathsf{P} \rangle i$	$\forall xy(xR_{\mathsf{F}}y \rightarrow yR_{\mathsf{P}}x)$
$@_i[\mathsf{P}]\langle \mathsf{F} \rangle i$	$\forall xy(xR_{\mathsf{P}}y \rightarrow yR_{\mathsf{F}}x)$

It follows that the conjunction $@_i[\mathsf{F}]\langle \mathsf{P} \rangle i \wedge @_i[\mathsf{P}]\langle \mathsf{F} \rangle i$ defines those frames in which R_{F} and R_{P} are mutually converse. And once we have this fundamental interaction defined, we can stop thinking in terms of separate R_{F} and R_{P} relations, instead viewing $\langle \mathsf{F} \rangle$ as looking forward along some binary relation $<$ (the "flow of time") and $\langle \mathsf{P} \rangle$ as looking backwards along the same relation. This enables us to define further temporally interesting properties:

$\langle \mathsf{P} \rangle \langle \mathsf{F} \rangle i$	$\forall xy \exists z(z < x \wedge z < y)$ *(Left-Directedness)*
$@_i(\langle \mathsf{F} \rangle \top \rightarrow \langle \mathsf{F} \rangle [\mathsf{P}][\mathsf{P}] \neg i)$	$\forall xy(x < y \rightarrow \exists z(x < z \wedge \neg \exists w(x < w < z)))$ *(Right-Discreteness)*

I mentioned in Example 2.1 that asymmetry was not definable in ordinary temporal logic. In fact, with the exception of the mutually converse property, transitivity, and density, none of the properties just defined are definable in orthodox temporal logic. Hybrid languages fill a genuine expressive gap when it comes to defining frames.

Remark 5.3 (All we need are satisfaction statements) *Note that if a formula φ defines a class of frames F, then so does the satisfaction statement $@_i\varphi$, where i is any nominal not occurring in φ. The relevance of this for tableaux will soon be clear.*

So nominals and @ enable us to define interesting classes of frames, and moreover every definable class of frames is definable using a satisfaction statement. This is pleasant — but the really important point is the way these frame defining powers interact with hybrid reasoning. Roughly speaking, if a pure formula α defines a class

of frames F, and we are free to introduce α as an axiom into our tableau proofs, then the axiom-enriched tableaux system is guaranteed to be complete with respect to F. For pure formulas, definability and completeness match perfectly.

More precisely, let A be a countable set of *pure satisfaction statements*, and $\mathsf{H}{+}\mathsf{A}$ be the tableau system that uses the formulas in A as axioms. That is, for any α in A, and any nominals $j, j_1, \ldots, j_n$ that occur on a branch of a tableau, we are free to add α or $\alpha[j_1/i_1, \ldots, j_n/i_n]$ to the end of that branch (here $i_1 \ldots, i_n$ are nominals in α, and $\alpha[j_1/i_1, \ldots, j_n/i_n]$ is the pure satisfaction statement obtained by uniformly substituting nominals for nominals as indicated).

Theorem 5.4 *Let A be a finite or countably infinite set of pure satisfaction statements, and let F be the class of frames that A defines (that is, the class of frames on which every formula in A is valid). Then $\mathsf{H}{+}\mathsf{A}$ is complete with respect to F.*

Proof. See Blackburn [13] for the unimodal case. The multimodal case is a straightforward generalization. ∎

Example 5.5 (An application in temporal logic) *Suppose we are working with the $\langle\mathsf{F}\rangle$ and $\langle\mathsf{P}\rangle$ temporal language, and that we are interested in models with a transitive flow of time. Which axioms guarantee completeness?*

The following suffice. First, to ensure that $\langle\mathsf{F}\rangle$ and $\langle\mathsf{P}\rangle$ really are mutually converse, add the axioms $@_i[\mathsf{F}]\langle\mathsf{P}\rangle i$ and $@_i[\mathsf{P}]\langle\mathsf{F}\rangle i$; we know from Example 5.2 that together these formulas define the converse property, and both are pure satisfaction statements. Now to guarantee transitivity. The pure formula $\langle\mathsf{F}\rangle\langle\mathsf{F}\rangle i \rightarrow \langle\mathsf{F}\rangle i$ defines this property. This is not a satisfaction statement, but $@_j(\langle\mathsf{F}\rangle\langle\mathsf{F}\rangle i \rightarrow \langle\mathsf{F}\rangle i)$ is, and this defines transitivity too.

What can we prove in this system? Here's an illustration. Note that for any choice of formula φ (not just pure formulas), $\langle\mathsf{P}\rangle\langle\mathsf{P}\rangle\varphi \rightarrow \langle\mathsf{P}\rangle\varphi$ is valid on the class of frames our axioms define. Thus, by Theorem 5.4, we should be able to prove any instance of this schema. And we can. In what follows i, j, and k, are chosen to be nominals not occurring in φ:

1			$\neg@_i(\langle\mathsf{P}\rangle\langle\mathsf{P}\rangle\varphi \rightarrow \langle\mathsf{P}\rangle\varphi)$		
2			$@_i\langle\mathsf{P}\rangle\langle\mathsf{P}\rangle\varphi$		$1, \neg\rightarrow$
$2'$			$\neg@_i\langle\mathsf{P}\rangle\varphi$		*Ditto*
3			$@_i\langle\mathsf{P}\rangle j$		$2, \langle\mathsf{P}\rangle$
$3'$			$@_j\langle\mathsf{P}\rangle\varphi$		*Ditto*
4			$@_j\langle\mathsf{P}\rangle k$		$3', \langle\mathsf{P}\rangle$
$4'$			$@_k\varphi$		*Ditto*
5			$@_j[\mathsf{P}]\langle\mathsf{F}\rangle j$		*Axiom*
6			$@_k\langle\mathsf{F}\rangle j$		$4, 5, [\mathsf{P}]$
7			$@_i[\mathsf{P}]\langle\mathsf{F}\rangle i$		*Axiom*
8			$@_j\langle\mathsf{F}\rangle i$		$3, 7, [\mathsf{P}]$
9			$@_k(\langle\mathsf{F}\rangle\langle\mathsf{F}\rangle i \rightarrow \langle\mathsf{F}\rangle i)$		*Axiom*
10		$\neg@_k\langle\mathsf{F}\rangle\langle\mathsf{F}\rangle i$	|	$@_k\langle\mathsf{F}\rangle i$	$9, \rightarrow$
11	$6, 10, \neg\langle\mathsf{F}\rangle$	$\neg@_j\langle\mathsf{F}\rangle i$		$@_k[\mathsf{F}]\langle\mathsf{P}\rangle k$	*Axiom*
12		✠ $8, 11$ ✠		$@_i\langle\mathsf{P}\rangle k$	$10, 11, [\mathsf{F}]$
13				$\neg@_k\varphi$	$2', 12, \neg\langle\mathsf{P}\rangle$
				✠ $4', 13$ ✠	

Once again, it is best to think of this proof in terms of a little graph-building automaton: it stepwise generates a graph and shows (now with the help of the axioms) that there is no coherent way to decorate the resulting structure with information.

In effect, Theorem 5.4 tells us that we can analyze hybrid reasoning in terms of a basic proof engine (such as our tableau rules) together with an axiomatic theory (at least so long as the axiomatic theory is formulated using only *pure* formulas). This is the way things work in first-order logic, and the resemblance is not coincidental. First, recall that the Standard Translation for hybrid languages maps nominals to free first-order variables. It follows that any pure formula φ defines a first-order class of frames (namely the class defined by the universal closure of $ST(\varphi)$). Second, analogous theorems have been proved for various hybrid languages, and although the completeness proofs differ in many respects, they typically have one ingredient in common: they use nominals to integrate the standard first-order model construction technique (the use of Henkin constants) with the standard modal technique (canonical models). As a number of authors emphasize (in particular Bull [21], Passy and Tinchev [41], and Blackburn and Tzakova [20]), such proofs show that hybrid logic genuinely blends modal and classical ideas.

Remark 5.6 (Related work) *Many of the same technical themes (including an essentially identical model construction technique) can be found in Basin, Matthews, and Vigano's [7] approach to labelled deduction for orthodox modal languages. The links between their work and the hybrid tradition deserves further exploration (for a start, many of their proof-theoretical insights may generalize to hybrid languages). Other general completeness results covering first-order definable frame classes have been proved for hybrid languages, such as Demri's [23] extension of the modal Sahlqvist theorem for his nominal-driven temporal sequent system.*

But the emphasis on *first-order* aspects of hybrid logic also point to the limitations of the previous theorem: it doesn't cover *second*-order frame classes — and many such classes are definable with the aid of propositional variables.

Example 5.7 (Second-order frame classes) *By making use of mixed formulas (that is, formulas containing both nominals and ordinary propositional variables) we can define $\mathbb{Z}$, the integers in their usual order, up to isomorphism; this cannot be done in first-order logic.*

The key observation is due to van Benthem [8], who points out that the simple $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$ language can almost define $\mathbb{Z}$. As he notes, the formula

$$([\mathsf{P}]([\mathsf{P}]p \to p) \to (\langle \mathsf{P} \rangle [\mathsf{P}]p \to [\mathsf{P}]p)) \wedge ([\mathsf{F}]([\mathsf{F}]p \to p) \to (\langle \mathsf{F} \rangle [\mathsf{F}]p \to [\mathsf{F}]p))$$

(a bidirectional variant of the Löb formula used in modal provability logic) defines $\mathbb{Z}$ up to isomorphism on the class of strict total orders without endpoints (that is, this Löb variant is valid on a frame $(T, <)$ that is a strict total order without endpoints iff $(T, <)$ is isomorphic to $\mathbb{Z}$.)

But it follows from standard modal results that we can't define strict total order without endpoints using only propositional variables — and this is where nominals come to the rescue. We have already seen that there are (pure) formulas defining the mutual converse property of $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$, transitivity, irreflexivity and trichotomy. Furthermore, the formulas $\langle \mathsf{F} \rangle \top$ and $\langle \mathsf{P} \rangle \top$ ensure that there are no endpoints. So the conjunction of all these (pure) formulas defines the class of strict total orders without endpoints — and hence conjoining the Löb variant yields a (mixed) formula valid on precisely the frames isomorphic to $\mathbb{Z}$. In a similar way, using a mixed formula it is

possible to define $\mathbb{N}$*, the naturals in their usual order, up to isomorphism; see Blackburn [11] for details. The second-order aspects of hybrid languages deserve further study.*

The result has another limitation: it gives no computational information. While the basic satisfaction problem for hybrid languages is PSPACE-complete, adding further axioms can have a wide range of effects: they may lower the problem into NP, leave it in PSPACE or lift it to EXPTIME (see Areces, Blackburn and Marx [3] for examples of all three possibilities). Nor is it difficult to devise axioms which result in logics with undecidable satisfaction problems. So the previous result tells us nothing about proof search or termination: it simply draws attention to a group of logic which are well-behaved from the perspective of completeness theory. It may well be that proof-theoretical and computational insights from the labelled deduction and description logic communities have a role to play in analyzing these logics further.

6 Binding Nominals to States

From the perspective of the Standard Translation, adding nominals to a modal language is in effect to add free variables over states. This immediately suggest a further extension: why not *bind* these "free variables", thus giving ourselves access to even more expressive power? I'll give a brief sketch of such logics, and then turn to the issue that interests me here: why they are relevant to knowledge representation.

Example 6.1 (Losers, jerks, and politicians) *Let's jump into the realms of pop-psychology and define a loser to be someone with no self-respect. Now, we can't define this concept in the hybrid logics we have seen so far; the closest we get is:*

$$i \wedge \neg\langle \textsc{respect} \rangle i.$$

This says that a specific individual i lacks self-respect. But we want more: we want a formula that is true at precisely those nodes (individuals) which lack a reflexive RESPECT *arc. We can get what we want by binding i out:*

$$\exists x(x \wedge \neg\langle \textsc{respect} \rangle x).$$

This sentence is true at precisely those those nodes at which it is possible to bind x to the current state, but impossible to loop back to the current state via the RESPECT *relation.*

Two remarks. First, the idea of binding nominals to the current state is so important in hybrid logic that a special notation (namely $\downarrow$*) has been introduced for it. So the previous sentence would normally be written:*

$$\downarrow x.\neg\langle \textsc{respect} \rangle x.$$

Second, as these examples illustrate, orthodox variable notation (x, y, z, and so on) is usually used for bound nominals.

OK — let's now define a jerk to be an idiot who admires himself:

$$\mathsf{idiot} \wedge \downarrow x.\langle \textsc{admires} \rangle x.$$

This sentence is satisfied at precisely those nodes which (1) have the idiot property, and (2) from which it is possible to take a reflexive step via the ADMIRES *relation.*

Finally, let's define a politician as a smooth talker such that everyone he talks to mistrusts him:

$$\downarrow x.(\mathsf{smooth\text{-}talker} \wedge \forall y(\langle \textsc{talks-to} \rangle y \rightarrow \neg @_y \langle \textsc{trusts} \rangle x).$$

Note the way the $@_y$ switches the perspective from the node x (the politician) to his audience.

I won't give a precise definition of the syntax and semantics of hybrid languages with $\forall$ and $\exists$ here (you can find all this in Blackburn and Seligman [15, 16] or Blackburn and Tzakova [18, 19]). The previous examples tell you pretty much everything you need to know, and the discussion that follows should clarify things further.

Remark 6.2 (We now have first-order expressivity) *Our new hybrid logic is strong enough to express any first-order concept. Here's the Hybrid Translation from first-order representations to our new hybrid logic:*

$$\begin{array}{lcl} HT(x\mathsf{R}_\pi y) & = & @_x\langle\pi\rangle y \\ HT(\mathsf{P}x) & = & @_x p \\ HT(x = y) & = & @_x y \\ HT(\neg\varphi) & = & \neg HT(\varphi) \\ HT(\varphi \wedge \psi) & = & HT(\varphi) \wedge HT(\psi) \\ HT(\exists v\varphi) & = & \exists v HT(\varphi) \\ HT(\forall v\varphi) & = & \forall v HT(\varphi). \end{array}$$

But although we *can* jump straight up to full first-order power, we don't have to. For a start, the use of @ in the hybrid translation is *crucial*. If we work with the @-free sublanguage, binding nominals to states with $\exists$ and $\forall$ does *not* yield full first-order expressive power; for a counterexample, see Proposition 4.5 of Blackburn and Seligman [15]. Hybrid logic decomposes the action of the classical quantifiers into two subtasks: perspective-shifting (performed by @) and binding (performed by the hybrid binders $\exists$ and $\forall$).

Moreover, we've seen that there is a useful restricted form of these binders, namely $\downarrow$. Some recent papers have explored hybrid logics with a primitive $\downarrow$ binder (without $\exists$ or $\forall$), and it turns out that such logics *characterize* the notion of locality; see Areces, Blackburn, and Marx [4].

Remark 6.3 (But even local binding is complex) *Be warned: $\downarrow$ may seem simple, but it's not. Even without @ (let alone $\forall$ or $\exists$, which are obviously powerful) it has an undecidable satisfaction problem. A detailed analysis is given in Areces, Blackburn, and Marx [5].*

Why is this? The following result (taken from Blackburn and Seligman [15]) may help the reader see why local binding is so powerful. We'll see — using a spypoint argument — that a hybrid language containing $\downarrow$ and just a single diamond lacks the finite model property. Let SCID4 be the conjunction of the following formulas:

$$\begin{array}{ll} S & x \wedge \neg\langle\mathsf{R}\rangle x \wedge \langle\mathsf{R}\rangle\neg x \wedge [\mathsf{R}]\langle\mathsf{R}\rangle x \\ C & [\mathsf{R}][\mathsf{R}] \downarrow y.(\neg x \rightarrow \langle\mathsf{R}\rangle(x \wedge \langle\mathsf{R}\rangle y)) \end{array}$$

I	$[\mathsf{R}]\downarrow y.\neg\langle\mathsf{R}\rangle y$
D	$[\mathsf{R}]\langle\mathsf{R}\rangle\neg x$
4	$[\mathsf{R}]\downarrow y.\langle\mathsf{R}\rangle(x \wedge [\mathsf{R}](\langle\mathsf{R}\rangle(\neg x \wedge \langle\mathsf{R}\rangle y \rightarrow \langle\mathsf{R}\rangle y)))$

Note that these formulas are pure, and that $\downarrow x.SCID4$ is a sentence. Moreover, note that this sentence has at least one model. For let $(\omega, <)$ be the natural numbers in their usual order, and suppose $s \notin \omega$ (s is the spypoint). Let $\mathcal{N}^s$ be the model bearing a single binary relation R defined as follows: W is $\omega \cup \{s\}$, R is $< \cup\{(n,s),(s,n) : n \in \omega\}$, and the valuation V is arbitrary. Clearly $\mathcal{N}^s, s \Vdash\downarrow x.SCID4$.

Obviously $\mathcal{N}^s$ is an infinite model. In fact any model $\mathcal{M} = (W, R, V)$ for $\downarrow x.SCID4$ is infinite. For suppose $\mathcal{M}, s \Vdash\downarrow x.SCID4$. Let $B = \{b \in W : sRb\}$. Because S is satisfied, $s \notin B$, $B \neq \emptyset$, and for all $b \in B$, bRs. Because C is satisfied, if $a \neq s$ and a is an R-successor of an element of B then a is also an element of B. As I is satisfied at s, every point in B is irreflexive; as D is satisfied at s, every point in B has an R-successor distinct from s; and as 4 is satisfied, R is a transitive ordering of B. So B is an unbounded strict partial order, thus B is infinite, hence so is W. So the ability to bind locally really does give us the power to see a lot of structure. And this power leads to undecidability (we can use spypoints to gaze upon the representation of some undecidable problem, such as an unbounded tiling problem).

Thus nominal binding offers (lots!) of new representational power — but how do we reason?

Remark 6.4 ($\forall$ and $\exists$ have classical tableau rules) *To cope with hybrid logic enriched with $\forall$ and $\exists$, we add the following rules to our tableau system. Note their form: they are the classical tableaux rules for existential and universal quantifiers:*

$$\frac{\neg@_s\exists x\varphi}{\neg@_s\varphi[t/x]} \qquad \frac{@_s\exists x\varphi}{@_s\varphi[a/x]}$$

$$\frac{\neg@_s\forall x\varphi}{\neg@_s\varphi[a/x]} \qquad \frac{@_s\forall x\varphi}{@_s\varphi[t/x]}$$

(Important: recall that a stands for a new nominal.) But while the rules are essentially classical, don't forget that the underlying language is different (after all, $\forall$ and $\exists$ bind formulas!). So as well as being able to prove all the standard classical quantificational principles (for example, $\forall x(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \forall x\psi)$, where x does not occur free in φ) we can also prove intrinsically modal principles. For example, $\exists xx$ is valid (this says: it is always possible to bind a variable to the current state). We can prove it as follows:

1	$\neg@_i\exists xx$	
2	$\neg@_i i$	$1, \neg\exists$
3	$@_i i$	*Ref*
	✠ *2, 3* ✠	

These rules give us a complete deduction system for hybrid logic with $\forall$ and $\exists$. Moreover, Theorem 5.4 extends to these systems: adding pure axioms yields a system complete with respect to the class of frames the axioms define. As before, "pure" simply means "contains no propositional variables", so we are free to make use of $\forall$ and

$\exists$ in our axioms. It follows (with the help of the Hybrid Translation) that we have a general completeness result that covers any first-order definable class of frames. Rules for $\downarrow$ can be found in Blackburn [13].

It's time to turn to the link with knowledge representation. I'll approach this topic via James Allen's classic work on temporal representation.

Example 6.5 (Allen style representations) *The core of Allen's system is an orthodox first-order theory of interval structure to which the metapredicate* **Hold** *has been added:* $\mathbf{Hold}(\mathsf{P}, i)$ *asserts that property* P *holds at the interval* i*.*

Allen then goes on to elaborate his account of properties. He introduces function symbols suggestively named and, or, not, exists, *and* all, *for combining property symbols, together with axioms governing them: for example*

$$\mathbf{Hold}(\mathit{and}(\mathsf{P}, \mathsf{Q}), i) \leftrightarrow \mathbf{Hold}(\mathsf{P}, i) \ \wedge \ \mathbf{Hold}(\mathsf{Q}, i).$$

It's clear Allen wants an 'internal' logic of terms that mirrors the 'external' logic of formulas. To put it another way, although he represents properties using terms, he wants them to behave like formulas.

This aspect of Allen's system has been criticized (some of the axioms governing the logical functions are rather odd; furthermore, as the structure of property terms is never fully specified, it's rather unclear what can and cannot be done with them; see Turner [54] and Shoham [53]). But I'm not so much interested in the details as the general strategy — for this is now standard in AI.

For example, if you look at Russell and Norvig [48] (in particular, the discussion of ontological engineering in Chapter 8) you'll see that Allen's approach has been generalized into a multistep methodology: (1) start with a first-order language; (2) reify the language heavily (that is, treat categories as individuals); (3) add metapredicates; and (4) when handling temporal aspects of ontology, induce boolean structure on the terms by adding and axiomatizing the logical functions and, or, *and* not *(*exists *and* all *are not discussed).*

Why is the methodology pioneered by Allen so popular? In my view, the point is the following. Knowledge representation is ultimately about representing information in a usable form — and this means bringing a variety of information types into a precise framework in which it can be manipulated as flexibly as possible. In essence, Allen's strategy is to start with first-order logic (because it's well understood) and then to mould it to the requirements of knowledge representation. Heavy use of reification and metapredicates allows general statements about a wide range of category types to be made. Logical functions are an attempt to soften the rigid distinction first-order logic draws between terms (which code referential information) and formulas (which code other types of information), thereby making more flexible representations possible. It's an interesting strategy — but it's *not* the only one.

Why not *start* with the intuition that all types of information should be treated democratically — or more accurately, *polymorphically*? This is the intuition behind hybrid logic. Hybrid logic begins with the observation that we *can* freely combine referential and non-referential information if we represent both types of information as *formulas*. Because this is our starting point, we don't need to introduce special logical functions and axioms to govern them — there is no term/formula distinction:

the standard connectives are responsible for combining all information right from the start. (Note that $@_i(p \wedge q) \leftrightarrow @_i p \wedge @_i q$, the hybrid analog of Allen's axiom for the *and* function, isn't something extra that needs to be stipulated: it's just a validity of hybrid logic, and can easily be proved in the basic tableau system.) Nor is there any mystery about what "property terms" are: Allen seems to have wanted properties to have a formula-like structure, and of course, that's *exactly* the form all representations take in hybrid logic. And binding nominals with $\forall$ and $\exists$ (which seems to correspond to Allen's intentions regarding the logical functions *exists* and *all*) will take us all the way up to first-order expressivity (if that's where we want to go).

7 The Sorting Strategy

In horticulture, hybrids are crossbreeds between distinct but related strains: ideally they combine the desirable properties of the parent strains in interesting new ways. Hybrid logic is certainly hybrid in this sense. Enriching modal logic with nominals and @ leads to systems that draw on both modal and first-order logic: we retain the locality and decidability of modal logic, gain the ability to name states and reason about their identity and their interrelationships, and (via nominal binding) open a novel route to first-order expressivity.

But hybrid logic is also a *sociological* hybrid: it's a meeting place for ideas from many traditions. We've seen that feature logic, description logic, and labelled deduction have independently developed key ideas of hybrid logic, and I've argued that the Allen-style ontological engineering languages can be viewed as strong hybrid languages. In short, a number of research communities, faced with similar problems (how best to represent and reason about graphlike structures) have come up with similar answers independently. Not only do they draw (consciously or unconsciously) on modal logic, they even moved beyond the barriers of modal orthodoxy in much the same way — the way encapsulated in hybrid logic.

But there is a third sense in which hybrid languages are hybrid, and this is perhaps the most important of all: hybrid languages are *intrinsically* hybrid. They allow us combine different sorts of information in a single formalism. In a nutshell, hybrid logics are *sorted modal logics*.

The importance of sorting has long been recognized in AI, linguistics, and philosophy: knowing that a piece of information is of a particular kind may allow us to draw useful conclusions swiftly and easily. But sorting has been neglected in the logical tradition: many useful kinds of sortal reasoning (for example, chaining through an inheritance hierarchy) are regarded as too simple to be of logical interest, and every logician knows that sorted first-order languages offer no new expressive power.

But sorted *modal* languages certainly do. As we have seen, by adding a second sort of atomic formula (nominals) and a new construct to exploit it (satisfaction operators), we can describe models in more detail and define new classes of frames. Moreover, we can create a basic reasoning system that is modally natural and supports a wide range of richer logics. But the hybrid languages of this paper have been simple two-sorted systems. Why stop there?

Example 7.1 (Sorting and fine-grained temporal reference)
Blackburn [12] presents multisorted modal logics with atomic formulas ranging over intervals of different lengths (seconds, hours, years, ...). This lets us build repre-

sentations like

$$\langle \mathsf{P} \rangle (3.05 \wedge \mathsf{P.M.} \wedge \mathsf{Friday} \wedge 26\mathsf{th} \wedge \mathsf{March} \wedge 1999 \wedge \mathsf{Vincent\text{-}accidentally\text{-}squeeze\text{-}the\text{-}trigger}),$$

which locates the trigger-squeezing event at the specific day and time the notation suggests. These logics are then extended to deal with indexical expressions (such as now, yesterday, today, and tomorrow), enabling us to build representations such as

$$\langle \mathsf{P} \rangle (\mathsf{Yesterday} \wedge \mathsf{Marvin's\text{-}head\text{-}explode}),$$

which locates the exploding-head event yesterday. Doing this properly means we have to sort two-dimensional modal logic (among other things, we need to guarantee that $\langle \mathsf{F} \rangle (\mathsf{yesterday} \wedge \varphi)$ *is false at every state in every model, for yesterday always lies in the past), and sorting turns out to be an effective way of exploiting two-dimensional semantics. The resulting logics are decidable (in fact, NP-complete) in many cases of interest.*

Example 7.2 (Sorting and paths) *When reasoning about branching time we often want to assert that that some event will take place in all possible paths into the future. This cannot be done in the temporal language in* $\langle \mathsf{F} \rangle$ *and* $\langle \mathsf{P} \rangle$*, even with the help of nominals and @.*

Bull [21] solved this problem by further sorting. He introduced a three-sorted modal language: in addition to propositional variables and nominals, his language contained path nominals, atomic formulas true at precisely the points on some path through a frame. He allowed explicit quantification over path nominals, and hence could define a "true at some state in every future" modality:

$$\langle \textsc{every-fut} \rangle \varphi \ := \ \forall \rho (\rho \rightarrow \langle \mathsf{F} \rangle \exists x (x \wedge \rho \wedge \varphi)).$$

Here ρ *is a bound path nominal, and* x *a bound nominal, so this says that on every path* ρ *through the current state, there is some future state* x *at which* φ *is true. See Goranko [32] and Blackburn and Tzakova [20] for more on hybrid languages for paths.*

I believe such examples point the way to an interesting line of work: dealing with *all* ontological distinctions in multisorted modal languages. At present little is known about what can and cannot be done in such systems, but interesting questions abound. I hope some equally interesting answers will soon be forthcoming.

A Brief Guide to the Literature

I have said little about the history of hybrid logic; these notes are an attempt to put this right, and provide a route into the hybrid literature. I'll omit references to applications of hybrid logic (such as feature logic) as these were given in the main text.

Hybrid logic was invented by Arthur Prior, the inventor of $\langle \mathsf{F} \rangle$ and $\langle \mathsf{P} \rangle$ based temporal logic (that is, *tense logic*). The germs of the idea seem to have emerged in discussion with C.A. Meredith in the 1950s, but the first detailed account is in Chapter V and Appendix B3 of Prior's 1967 book *Past, Present, and Future* [42]. Several of the papers collected in *Paper on Time and Tense* [43] allude to or discuss hybrid languages, and the posthumously published book *Worlds, Times and Selves* [44]

is solely devoted to the topic (unfortunately, the book is only an approximation to Prior's intentions: it's essentially a reconstruction, by Kit Fine, of notes found after Prior's death in 1969). Prior called nominals *world propositions*, typically worked with very rich hybrid languages (he bound nominals using $\forall$ and $\exists$) and made heavy use of near-atomic satisfaction statements like the ones used in our tableau systems.

The next big step was Robert Bull's 1970 paper "An Approach to Tense Logic" [21]. Bull introduced a three-sorted hybrid language (propositional variables, nominals, and path nominals), noted that the presence of $\forall$ and $\exists$ made it easy to combine the modal canonical model construction with the first-order Henkin construction (and thus proved the earliest version of Theorem 5.4), and re-thought modal and hybrid completeness theory in terms of Robinson's non-standard set theory. It's a (too long overlooked) classic. Tough going in places, it repays careful reading.

I know of no more papers on the subject till the 1980s, when hybrid logic was independently reinvented by a group of Bulgarian logicians (Solomon Passy, Tinko Tinchev, George Gargov, and Valentin Goranko). The locus classicus of this work is Passy and Tinchev's "An Essay on Combinatoric Dynamic Logic" [41], a detailed study of hybrid Propositional Dynamic Logic. Like Bull's paper, it's one of the must reads of the hybrid literature (but don't overlook the many other excellent papers by these authors, such as [40, 40, 30, 28, 29].) The Sofia School did discuss nominal binding with $\forall$ and $\exists$, but one of their enduring legacies is that they initiated the study of binder-free systems. Gargov and Goranko's "Modal Logic with Names" [27] studies such systems in the setting of unimodal logic, and my own "Nominal Tense Logic" [11] does so in tense logic.

During the 1990s, the emphasis has been on understanding the hybrid hierarchy in more detail. Goranko [31] introduced $\downarrow$, Blackburn and Seligman [15, 16] examined the interrelationships between a number of different binders, and Blackburn and Tzakova [18, 20] mapped hybrid completeness theory for many of these systems. Intuitions about locality hinted at in some of these papers are placed on a firm mathematical footing in Areces, Blackburn and Marx [4]; the paper also proves some fundamental interpolation and complexity results (see also [5], by the same authors, for a detailed discussion of undecidability in $\downarrow$ based logics). The late 1990's also saw a number of papers of hybrid proof theory: Blackburn [13], Demri [23], Demri and Goré [24], Konikowska [36], Seligman [52] and Tzakova [55]. Actually, pioneering work had been done by Seligman at the beginning of the decade (see [50, 51]); unfortunately his work was overlooked.

Here's three suggestions for further reading. First, Chapter 7 of Blackburn, de Rijke, and Venema [14] contains a textbook level discussion on how to blend the canonical model and Henkin constructions (the idea behind Theorem 5.4 and its analogs). Second, "Complexity Results for Hybrid Temporal Logics" [3] a recent paper by Areces, Blackburn and Marx studies complexity issues in some detail. The proofs make heavy use of relational structures and have a strong geometric content. The paper relates the results to issues in temporal (and, in spite of the title, description) logic; for many readers this would be a good place to learn more about the expressivity hybrid languages offer. Third, Marx [38] is a review of *HyLo'99* (the First International Workshop on Hybrid Logic). This will give you a birds-eye-view of current issues in the field. In addition, Carlos Areces has recently created a hybrid logic website at `http://www.illc.uva.nl/~carlos/hybrid`. You can find the papers just mentioned

(and others) there.

Acknowledgments

In writing this paper I have tried to remain faithful to the spirit of my *M4M* talk, which means I have drawn on many discussions of modal logic, relational structures, hybrid logic, and related topics. I particularly wish to thank Carlos Areces, Johan van Benthem, Valentin Goranko, Edith Hemaspaandra, Maarten Marx, Maarten de Rijke, Jerry Seligman, Miroslava Tzakova, and Yde Venema, who have deeply influenced the way I think about these topics. I am also grateful to the participants of *M4M* and *HyLo'99* for interesting discussion, and to David Basin and Luca Viganò for helping me understand labelled deduction better. Thanks to Claire Gardent and Ewan Klein for their comments on the first version. Finally, a big thank you to the two referees for their detailed comments, and to Carlos Areces for his painstaking editorial help.

References

[1] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, July 1984.

[2] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[3] C. Areces, P. Blackburn, and M. Marx. Complexity results for hybrid temporal logics. To appeal in *Logic Journal of the IGPL*, 1999.

[4] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. Technical Report CLAUS-Report 104, Computerlinguistik, Universität des Saarlandes, 1999. `http://www.coli.uni-sb.de/cl/claus`. To appear in *Journal of Symbolic Logic*.

[5] C. Areces, P. Blackburn, and M. Marx. A roadmap on the complexity of hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999.

[6] C. Areces and M. de Rijke. Accounting for assertional information. Manuscript, 1999.

[7] D. Basin, S. Matthews, and L. Viganò. Labelled propositional modal logics: theory and practice. *Journal of Logic and Computation*, 7:685–717, 1997.

[8] J. van Benthem. *The Logic of Time*. Kluwer Academic Publishers, Dordrecht, second edition, 1991.

[9] S. Bird and P. Blackburn. A logical approach to arabic phonology. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–29, 1991.

[10] P. Blackburn. Modal logic and attribute value structures. In M. de Rijke, editor, *Diamonds and Defaults*, Synthese Language Library, pages 19–65. Kluwer Academic Publishers, Dordrecht, 1993.

[11] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 14:56–83, 1993.

[12] P. Blackburn. Tense, temporal reference, and tense logic. *Journal of Semantics*, 11:83–101, 1994.

[13] P. Blackburn. Internalizing labelled deduction. Technical Report CLAUS-Report 102, Computerlinguistik, Universität des Saarlandes, 1998. `http://www.coli.uni-sb.de/cl/claus`. To appear in *Journal of Logic and Computation*.

[14] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. 1999. A draft version of this book is available at `http://www.coli.uni-sb.de/cl/claus`.

[15] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Special issue on decompositions of first-order logic.

[16] P. Blackburn and J. Seligman. What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 1, pages 41–62. CSLI Publications, Stanford University, 1998.

[17] P. Blackburn and E. Spaan. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2:129–169, 1993.

[18] P. Blackburn and M. Tzakova. Hybrid completeness. *Logic Journal of the IGPL*, 6:625–650, 1998.

[19] P. Blackburn and M. Tzakova. Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, 24:23–49, 1998.

[20] P. Blackburn and M. Tzakova. Hybrid languages and temporal logics. *Logic Journal of the IGPL*, 7(1):27–54, 1999.

[21] R. Bull. An approach to tense logic. *Theoria*, 36:282–300, 1970.

[22] G. De Giacomo. *Decidability of class-based knowledge representation formalisms*. PhD thesis, Universitá di Roma "La Sapienza", 1995.

[23] S. Demri. Sequent calculi for nominal tense logics: a step towards mechanization? In N. Murray, editor, *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, volume 1617 of *LNAI*, pages 140–154. Springer Verlag, 1999.

[24] S. Demri and R. Goré. Cut-free display calculi for nominal tense logics. In N. Murray, editor, *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, volume 1617 of *LNAI*, pages 155–170. Springer Verlag, 1999.

[25] M. Fitting. *Proof Methods for Modal and Intuitionistic Logic*. Reidel, 1983.

[26] D. Gabbay. *Labelled Deductive Systems*. Clarendon Press, Oxford, 1996.

[27] G. Gargov and V. Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22:607–636, 1993.

[28] G. Gargov and S. Passy. Determinism and looping in combinatory PDL. *Theoretical Computer Science*, 61:259–277, 1988.

[29] G. Gargov and S. Passy. A note on Boolean modal logic. In P. Petkov, editor, *Mathematical Logic. Proceedings of the 1988 Heyting Summerschool*, pages 311–321. Plenum Press, New York, 1990.

[30] G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, editor, *Mathematical Logic and its Applications*, pages 253–263. Plenum Press, 1987.

[31] V. Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.

[32] V. Goranko. An interpretation of computational tree logics into temporal logics with reference pointers. Technical Report Verslagreeks van die Department Wiskunde, RAU, Nommer 2/96, Department of Mathematics, Rand Afrikaans University, Johannesburg, South Africa, 1996.

[33] V. Goranko and S. Passy. Using the universal modality: Gains and questions. *Journal of Logic and Computation*, 2:5–30, 1992.

[34] J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the Association for Computing Machinery*, 38:935–962, 1991.

[35] W. Hodges. *Model Theory*. Cambridge University Press, Cambridge, 1993.

[36] B. Konikowska. A logic for reasoning about relative similarity. *Studia Logica*, 58:185–226, 1997.

[37] M. Kracht. Power and weakness of the modal display calculus. In H. Wansing, editor, *Proof Theory of Modal Logic*, pages 93–121. Kluwer Academic Publishers, Dordrecht, 1996.

[38] M. Marx. First International Workshop on Hybrid Logic (HyLo '99). *Logic Journal of the IGPL*, 7:665–669, 1999. Conference Report.

[39] M. Marx and Y. Venema. *Multidimensional Modal Logic*, volume 4 of *Applied Logic Series*. Kluwer Academic Publishers, Dordrecht, 1997.

[40] S. Passy and T. Tinchev. Quantifiers in combinatory PDL: completeness, definability, incompleteness. In *Fundamentals of Computation Theory FCT 85*, volume 199 of *LNCS*, pages 512–519. Springer, 1985.

[41] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[42] A. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[43] A. Prior. *Papers on Time and Tense*. University of Oxford Press, 1968.

[44] A. Prior and K. Fine. *Worlds, Times and Selves*. University of Massachusetts Press, 1977.

[45] M. Reape. An introduction to the semantics of unification-based grammar formalisms. Technical Report DYANA deliverable R3.2.A, ESPRIT basic research action BR 3175, Center for Cognitive Science, University of Edinburgh, 1991.

[46] M. Reape. A feature value logic. In C. Rupp, M. Rosner, and R. Johnson, editors, *Constraints, Language and Computation*, Synthese Language Library, pages 77–110. Academic Press, 1994.

[47] M. de Rijke. The modal logic of inequality. *Journal of Symbolic Logic*, 57:566–584, 1992.

[48] S. Russell and P. Norvig. *Artificial Intelligence*. Prentice Hall, 1995.

[49] K. Schild. A correspondence theory for terminological logics. In *Proceedings of the 12th IJCAI*, pages 466–471, 1991.

[50] J. Seligman. A cut-free sequent calculus for elementary situated reasoning. Technical Report HCRC-RP 22, HCRC, Edinburgh, 1991.

[51] J. Seligman. Situated consequence for elementary situation theory. Technical Report Logic Group Preprint IULG-92-16, Indiana University, 1992.

[52] J. Seligman. The logic of correct description. In M. de Rijke, editor, *Advances in Intensional Logic*, pages 107–135. Kluwer, 1997.

[53] Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. The MIT Press, Cambridge, MA, 1988.

[54] R. Turner. *Logics for Artificial Intelligence*. Ellis Horwood: Chichester, 1984.

[55] M. Tzakova. Tableaux calculi for hybrid logics. In N. Murray, editor, *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, volume 1617 of *LNAI*, pages 278–292. Springer Verlag, 1999.

Received February 10, 2000

Interest Group in Pure and Applied Logics (IGPL)

The Interest Group in Pure and Applied Logics (IGPL) is sponsored by The European Association for Logic, Language and Information (FoLLI), and currently has a membership of over a thousand researchers in various aspects of logic (symbolic, mathematical, computational, philosophical, etc.) from all over the world (currently, more than 50 countries). Our main activity is that of a research and information clearing house.

Our activities include:

- Exchanging information about research problems, references and common interest among group members, and among different communities in pure and applied logic.
- Helping to obtain photocopies of papers to colleagues (under the appropriate copyright restrictions), especially where there may be difficulties of access.
- Supplying review copies of books through the journals on which some of us are editors.
- Helping to organise exchange visits and workshops among members.
- Advising on papers for publication.
- Editing and distributing a Newsletter and a Journal (the first scientific journal on logic which is FULLY electronic: submission, refereeing, revising, typesetting, publishing, distribution; first issue: July 1993): the Logic Journal of the Interest Group on Pure and Applied Logics. (For more information on the Logic Journal of the IGPL, see the Web homepage: http://www.jigpal.oupjournals.org)
- Keeping a public archive of papers, abstracts, etc., accessible via ftp.
- Wherever possible, obtaining reductions on group (6 or more) purchases of logic books from publishers.

If you are interested, please send your details (name, postal address, phone, fax, e-mail address, research interests) to:

IGPL Headquarters
c/o Prof. Dov Gabbay
King's College, Dept of Computer Science
Strand
London WC2R 2LS
United Kingdom
e-mail: dg@dcs.kcl.ac.uk

For the organisation, Dov Gabbay, Ruy de Queiroz and Hans Jürgen Ohlbach