Generating Hard Modal Problems for Modal Decision Procedures

Ian Horrocks
Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

Peter F. Patel-Schneider
Bell Labs Research
Murray Hill, NJ, U.S.A.
pfps@research.bell-labs.com

Abstract

Random generation of modal formulae is a viable method for generating problems for benchmarking modal decision procedures. However, previous work in this area has used a flawed generator that has resulted in questionable results. Fixing the generator changes the characteristics of the generated problems. The fixed generator can be used to generate hard problems that have more interesting modal properties than previous hard problem sets.

Optimised decision procedures for propositional modal logics have recently been developed. These decision procedures are significantly faster than previous decision procedures for modal logics, to the extent that some problems that were impossible to solve using older decision procedures are trivial for the newer decision procedures. Further, due to the connection between propositional modal logics and description logics, there is a natural use for these optimised decision procedures in description logic classifiers.

Now that such modal decision procedures are available, it is possible, and even necessary, to determine their actual performance, both relative and absolute, on hard, interesting modal problems. By hard problems, we mean problems that are hard, but not impossible, for one or, preferably, more current decision procedures, not necessarily problems that are theoretically hard. Several mechanisms have been proposed for generating such hard modal problems. These mechanisms generally break down into three groups: 1/ hand generation of classes of modal formulae, 2/ translation of actual problems into modal formulae, and 3/ random generation of hard modal formulae.

The first group is best illustrated by the modal formulae used in the Tableaux'98 comparison of modal decision procedures [Heuerding and Schwendimann, 1996; Balsiger and Heuerding, 1998]. For this comparison, 54 parameterised formulae generators were created. Each generator took a size parameter and output a formula whose difficulty of proving (or equivalently determining the satisfiability of) was supposed to increase exponentially with the size of the parameter.

The second group is illustrated by the translation of description logic knowledge bases, such as the GALEN knowledge base [Rector *et al.*, 1994], or other knowledge descriptions, such as entity-relationship diagrams [Calvanese *et al.*, 1998], into propositional modal logics. This mechanism was used in a comparison of description logic systems at DL'98 [Horrocks and Patel-Schneider, 1998a].

The third group is illustrated by the work of Giunchiglia and Sebastiani [Giunchiglia and Sebastiani, 1996a; 1996b] and Hustadt and Schmidt [Hustadt and Schmidt, 1997a;

1997b]. This work uses parameterised random generators to create collections of hard (or at least large and non-trivial) modal formulae in the logic $\mathbf{K}_{(\mathbf{m})}$.

Even considering the above efforts, there is a general lack of hard modal problems on which to test modal decision procedures. This is in contrast to the situation in non-modal propositional modal logics. There, because of the continuing use of propositional reasoners to solve problems from chip layout, planning, *et cetera*, there is a large collection of actual problems. These problems are hard (and/or large) enough to stress even the decision procedures being currently developed. Also, there is general agreement on a generator for hard random formulae [Selman *et al.*, 1996]. Hand generation is not widely used, as it is too hard to generate interesting problems.

In modal logics there are very few real problems that can be used as sources of hard modal problems. This is, in part, due to lack of optimised decision procedures for modal logics—if there are no fast modal decision procedures, there is no incentive to map problems into modal logics. This situation may change in the future as the result of the availability of the current group of optimised modal decision procedures. Further, random generation of hard modal formulae is much more difficult than random generation of hard propositional formulae due to the larger number of parameters needed to define their structure.

Hustadt and Schmidt's experiments led them to conclude that, for a given size, the hardest modal formulae have a very simple structure, with a modal depth of 1 and propositional variables occurring only at depth 1. (A purely propositional formula is said to have a depth of 0.) Even if their conclusions are valid, and we take issue with them at least in part, it is not a good idea to test modal decision procedures only on formulae that fall into this very restricted class. A modal decision procedure that is very much faster than other modal decision procedures on this particular kind of formulae may be very much slower than other modal decision procedures on other formulae. Further, formulae of this form will not correspond to the sort of formulae involved in real modal problems, so comparisons on them will be of little use in predicting behaviour with real problems.

What is needed is a good, systematic mechanism to generate hard problems with real, varying modal content. Again, we mean problems that are hard for current modal decision procedures. This mechanism should have parameters to control the amount, form, and depth of modal content. It should also reliably generate problems of a given hardness, with few or no trivial problems. Such a mechanism could be used to compare the current optimised modal decision procedures.

To this end, we analyse the generators designed by Giunchiglia and Sebastiani and Hustadt and Schmidt, and show how problems with these generators led to a serious underestimation of the impact of modal depth on the hardness of the problems generated. We propose a modification of these generators that produces much harder modal problems, and use the modified generator to compare various decision procedures and optimisations. Finally, we analyze the results of this testing.

Note that creating a good mechanism for generating hard random problems is independent of whether is it a good idea to use random problems at all. We are of the firm belief that random problems, at least random problems that are not related to actual problems, are not the best way to test decision procedures. However, in the absence of actual problems for testing modal decision procedures, we are reduced to using random problems. Perhaps the performance of the new decision procedures on random problems will encourage the use of the new decision procedures on actual problems, which can then be used as benchmarks for the decision procedures or as models for the generation of such benchmarks.

Techniques for Building Modal Decision Procedures

Building a decision procedure for a simple modal logic is not much more difficult than building a decision procedure for propositional logic. It suffices to take any tableau expansion methodology for the modal logic and directly implement that methodology, using a simple search to deal with the inherent nondeterminism. Of course, the resulting system will be completely unusable, as it will reflect all the inefficiencies of tableau expansion.

There are several ways of producing faster modal decision procedures. One method is to treat a modal proof as a collection of non-modal propositional proofs. This method, as exemplified in the KSAT decision procedure [Giunchiglia and Sebastiani, 1996a; 1996b], treats the modal sub-formulae as proposition atoms and performs a satisfiability check on the resulting formula. If any propositional model is found for this formula, the true modal formulae are extracted from the model and for each modal successor the process is repeated. If all the modal checks succeed, a complete modal model has been found; if they fail, the decision procedure tries other propositional models until all possibilities have been exhausted.

The advantage of this method is that it can use a state of the art optimised propositional decision procedure, gaining the speed advantages of that decision procedure. Only minor changes are needed in the propositional decision procedure to support modal reasoning. The disadvantage of this method is that the different propositional steps are only loosely connected to the modal steps, and information gained in the modal steps may not be available to improve the overall performance of the decision procedure, which is dominated by the propositional steps.

Another method for obtaining a fast modal decision procedure is to translate modal formulae into first-order formulae, which are then checked using a first-order prover. This approach is used in the TA system [Hustadt and Schmidt, 1997a; 1997b]. At first glance this may seem to be a poor approach as first-order logic is undecidable. However, it is possible to translate modal formulae into a decidable fragment of first-order logic (FOL), and when combined with a first-order prover that is sound and complete for the fragment in question this gives a sound and complete decision procedure for the modal logic.

This method again has the advantage that it can use a state of the art first-order prover, selecting one which is either naturally fast on the kinds of formulae produced by the translation, or which can be tuned to be fast on these formulae. One disadvantage of this mechanism is that it is hard to control first-order provers, so performance may be poor on some formulae.

A third method for obtaining a fast modal decision procedure is to build an optimised decision procedure from scratch, using whatever optimisations are effective for modal logics. Two decision procedures built using this method are FaCT [Horrocks, 1998] and DLP [Patel-Schneider, 1998]. Both decision procedures were actually built as description logic systems, but, because of the relationships between description logics and modal logics, include a modal logic decision procedure.

The advantage of this third method is that the decision procedure can be optimised for propositional modal logics, and does not depend on optimisations designed for other logics. The disadvantage of this method is that the decision procedure has to be built from scratch and thus does not automatically get the benefits of optimisation work in propositional or first-order logics.

However an optimised modal decision procedure is built, its core is still a search engine that explores the space of potential proofs, or refutations, or models for a formula. Although there are techniques that can be used to avoid search, such as normalising the input formula to detect local inconsistencies and tautologies, the search engine's performance is vital to the overall performance of the decision procedure. Further, there are a number of differing approaches that result in performance variations on different types of formulae.

On Being a Good Adversary

The biggest problem in generating problem sets for modal decision procedures is to strike the right balance. If a problem set is too easy, it will not show off sophisticated optimisations that have a high overhead, as even less-optimised decision procedures will complete the problems quickly. If a problem set is too hard, the decision procedures will not terminate within the time limit set for the test, and their effectiveness cannot be determined. If a problem set spans only a small part of the problem space, then it may concentrate on or miss areas where a decision procedure is relatively fast or slow. Ideally, then, a problem set should have some easy sections and some hard ones, and should contain a large variety of problems.

It is possible that a large formula can have subformulae that are tautologous or contradictory. Even if these subformulae do not make the entire formula tautologous or contradictory, the presence of these subformulae may mean that the formula is not an effective test.

In the propositional case, generating formulae with the appropriate characteristics is now relatively easy. The input form can be restricted to conjunctive normal form with 3 literals per clause (3CNF), as it is generally agreed that this restriction does not make the problem any simpler [Selman *et al.*, 1996]. Each clause can be generated by randomly selecting a *combination* of three propositional atoms and negating each of them with probability one-half. There is a general agreement that the hardness of a problem set in this form is determined by the number of propositional atoms and the number of clauses. In particular, to get a mix of interesting problems—some satisfiable, some unsatisfiable, some relatively easy, some relatively hard—it suffices to fix the number of propositional atoms and vary the number of clauses. With relatively few clauses almost all problems are satisfiable and easy; with relatively many clauses almost all problems are unsatisfiable and easy; with a critical number of clauses, about 4.2 times as many as the number of propositional variables, about half the problems are satisfiable and many of them are hard. Many experiments using this generation mechanism have been performed on various propositional satisfiability (SAT) decision procedures [Freeman, 1996; Selman *et al.*, 1996].

All well-designed SAT algorithms have been shown to exhibit a form of this easy-hard-easy behavior: for a given number of propositional variables, problems with a number of clauses that makes them either under-constrained ($\gg$50% satisfiable) or over-constrained ($\ll$50% satisfiable) are generally much easier to solve than critically constrained problems. This phenomenon has also been observed in a range of other NP-complete problems [Hogg *et al.*, 1996]. Moreover, for sufficiently large numbers of propositional variables, the transition from under- to over-constrained becomes rapid, forming a phase transition.

In the modal case, there are many more difficulties. First, even for formulae in a (generalised) 3CNF form, there are additional variations in the structure requiring parametric control: the "atoms" in clauses can be propositional literals or modal formulae, and if modal they contain a sub formula the structure of which can itself be varied. Second, it is not well understood how these additional parameters interact with the propositional parameters to determine the hardness of problems, and there is as yet no general agreement as to whether a phase transition can be observed in PSPACE-complete modal problems. Given our limited understanding of these issues, generators of modal problems must be carefully designed and rigorously tested.

Previous Problem-Generation Techniques

In order to test their KSAT decision procedure, Giunchiglia and Sebastiani developed a random generator for modal formulae that generalised the 3CNF model described in Section [Giunchiglia and Sebastiani, 1996a; 1996b]. The generator produces conjunctive formulae of the form $(D_1 \wedge \ldots \wedge D_L)$ where each D_i is a K-disjunctive formula of the form $(C_1 \vee \ldots \vee C_K)$. Each disjunct C_j can be either a literal (a propositional variable or its negation) or a (possibly negated) modal atom. Modal atoms are of the form $\Box_i D$ where i is one of the modalities and D is another K-disjunctive formula.

Generation is controlled by six parameters: N, the number of different primitive concepts (propositional variables); M, the number of different modalities; K, the size of the K-disjunctive formulae; D, the maximum modal depth; P, the probability of a disjunct being a literal rather than a modal atom (except at depth D); and L, the number of K-disjunctive formulae in the top-level conjunction. If P is 1, the formulae generated are purely propositional and, it was claimed by Giunchiglia and Sebastiani, are of the standard SAT testing form [Giunchiglia and Sebastiani, 1996b].

The experiments devised by Giunchiglia and Sebastiani were designed to test the performance of $\mathbf{K_{(m)}}$ decision procedures, and to discover if a phase transition could be observed. Three sets of experiments were performed by varying one of the parameters N, M and D while keeping the others fixed. The values of the fixed parameters were chosen so that varying L to give values L/N in the range 1–40 produced problems ranging from $\approx$100% satisfiable to $\approx$0% satisfiable. In all the experiments, K was fixed at 3 and P was fixed at 0.5.

We will mostly concern ourselves with the set of experiments in which the modal depth was varied, the values chosen being $D = 2$, 3, 4 and 5, with N fixed at 3 and M at 1; we will refer to these four parameter settings as **PS4**, **PS3**, **PS2** and **PS1** respectively, following Hustadt and Schmidt. The problems generated were much harder (for the KSAT decision procedure) if the value of N (the number of propositional variables) was increased, but became only slightly harder with increasing modal depth.

Hustadt and Schmidt [Hustadt and Schmidt, 1997a; 1997b] subsequently pointed out that the formulae produced by Giunchiglia and Sebastiani's generator contained many tautological and contradictory clauses which considerably reduced the effective size of the formulae and often rendered larger formulae trivially unsatisfiable. This exaggerated the benefits of the KSAT decision procedure as it performed particularly well with these kinds of formula. They also noted that with the other parameters fixed, increasing the modal depth greatly increased the size of the formulae generated. As it had already been shown that increasing the modal depth had little effect on the hardness of problems, they concluded that for a given size of formula increasing the modal depth actually made problems easier, and that the value of N (the number of propositional variables) was by far the most important factor in determining the hardness of the generated formulae. Having modified the generator so that far fewer tautological and contradictory clauses were produced (making for generally much harder problems)

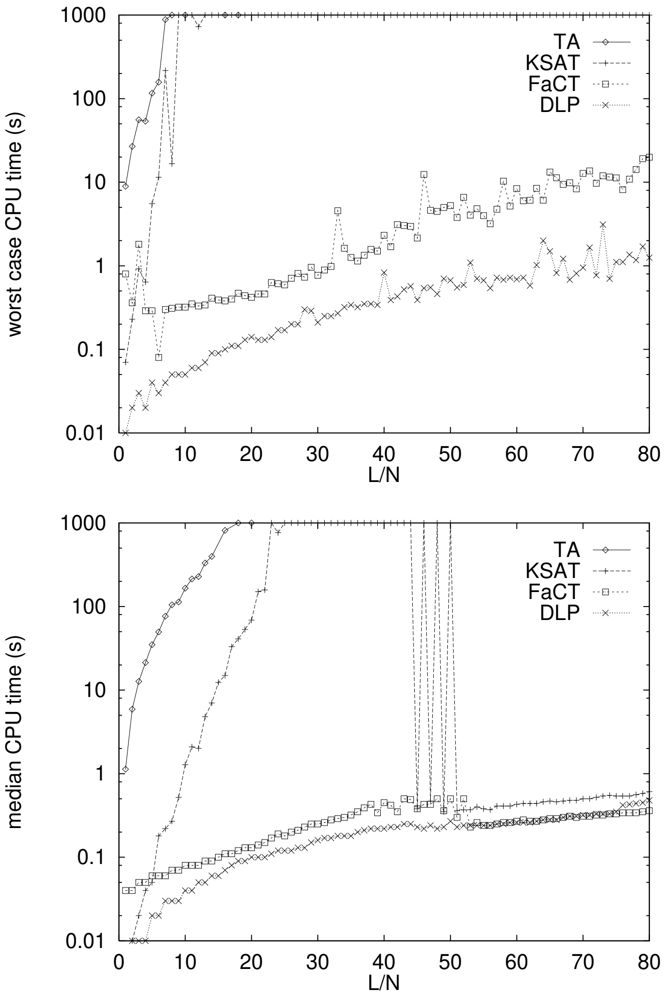

Figure 1: Worst-case and median CPU times for **PS1**

they went on to perform new experiments, their conclusion about the effect of increasing modal depth leading them to fix the value of this parameter at 1 in all cases.

Hustadt and Schmidt also improved the analysis of the experimental results by considering not just median times but a range of percentile times, the Nth percentile being the time sufficient to solve $N\%$ of the problems in the test set. We will mostly restrict our attention here to median (50th percentile) and worst-case (100th percentile) times. Worst-case times can be important in realistic applications if, for example, a real-time response is required or if they are so large that they dominate the average solution time.

Closer consideration reveals that the apparent reduction in the difficulty of problems with increasing modal depth is an artifact of the Giunchiglia and Sebastiani generator, and in particular its production of tautological propositional clauses such as $(P \vee \neg P \vee \ldots)$ that can be simplified to $\top$ (True). When this occurs inside a modal atom $\Box_i \top$ it can again be simplified to $\top$, and there is then a probability (usually fixed at 0.5) that the modal atom will be negated to give $\bot$ (False). If all the disjuncts in a K-

disjunctive formula are $\perp$, then the formula can be simplified to $\perp$, and if this occurs in one of the K-disjunctive formulae in the top level conjunction, then the whole formula can be simplified to $\perp$ without invoking the modal decision procedure. Increasing the modal depth increases the probability that such trivially unsatisfiable formulae will be generated.

Repeating the original experiments using the Hustadt and Schmidt generator gives much different results for larger values of D. Using the original generator, the median solution time even for the hardest problems generated by the **PS1** settings never exceeded 10s of CPU time for either KSAT or TA; with the modified generator, median solution times for both systems often exceeded the 1,000s maximum allowed in the experiment. Figure 1 shows the median and worst-case solution times for both systems, as well as those for FaCT and DLP, using the modified generator.

The large oscillations in the median solution times using KSAT with L/N in the range 45–51 reflect the fact that this is the region where the probability of problems being satisfiable is approximately 0.5, and like FaCT and DLP, KSAT can solve most unsatisfiable problems in this test relatively easily. TA is also able to solve some of the unsatisfiable problems in this region, although not as many as the other systems, but it was not possible to complete the experiment with TA as it does not deal reliably with the larger problems in the test.

Unsatisfiable formulae here are still mostly relatively easy for all the systems because they are unsatisfiable considering only their non-modal portion. As all the decision procedures are biased towards looking first at the non-modal information, they can quickly dispose of such formulae. Hustadt and Schmidt have called these formulae trivially unsatisfiable.

Examining more closely the results for DLP, the fastest decision procedure tested, shows that the median solution time for problems in the ≈50% satisfiable region (L/N in the range 40–50) increases from ≈0.05s for **PS4** to ≈0.25s for **PS1**: Figure 2 shows the probability of generating satisfiable formulae and DLP's median satisfiability times for all four tests. The median size (measured syntactically) of **PS1** formulae is approximately eight times that of **PS4** formulae.

In recent work, Giunchiglia *et al* [Giunchiglia *et al.*, 1998] have further improved the generator, but their testing concentrated solely on formulae with $D = 1$ and $P = 0$.

A Modified Problem-Generation Technique

The formulae produced by Hustadt and Schmidt's generator are generally harder, and clearly demonstrate performance differences between the various systems, but they are still far from ideal for testing modal decision procedures: The formulae are very similar to each other, and this can exaggerate the importance of a particular reasoning or optimisation technique. There are still relatively few hard problems produced, particularly unsatisfiable ones. For a given value of L, the probability of generating unsatisfiable formulae decreases for larger modal depths, and achieving a balance between satisfiable and unsatisfiable tests requires very large formulae to be generated. As modal depth increases, the increase in hardness does not match the increase in formula size.

To overcome some of these problems, we have further modified the generator in order to produce much less uniform formulae. Such formulae have the benefit of being generally much harder (for a given formula size), and of being much more likely to be unsatisfiable. Our generator has additional parameters K_{min} and D_{min} that respectively define the minimum size of disjunctive expressions and the minimum modal

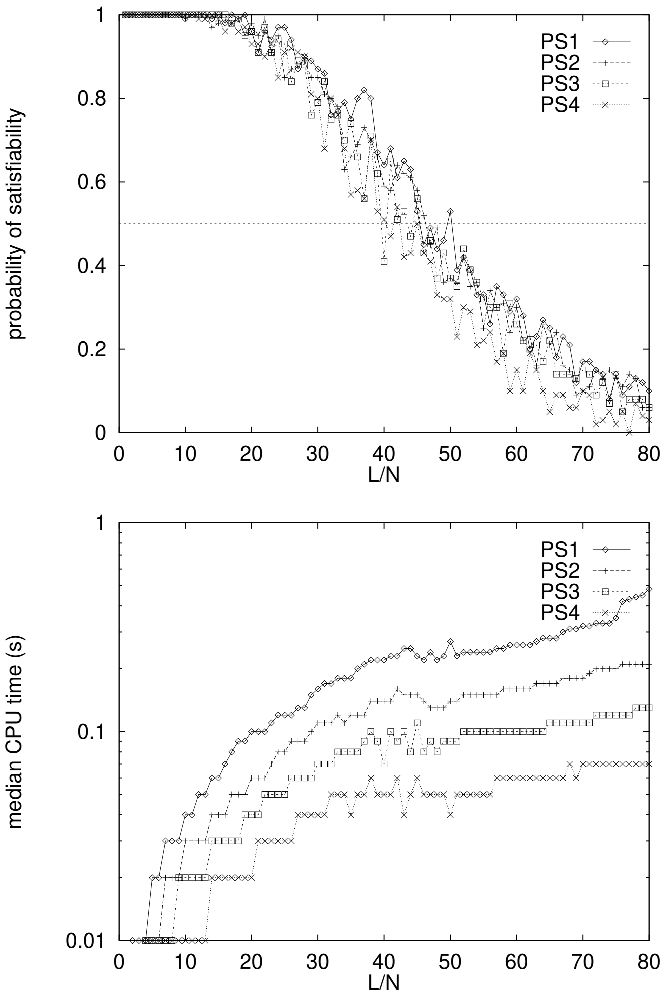

Figure 2: Median CPU times for DLP with **PS1-4**

depth. The values occurring within a formula are then varied at random between the specified maxima and minima.

With the parameters set at $N = 6$, $M = 1$, $P = 0.5$, $K_{min} = 1$, $K_{max} = 4$, $D_{min} = 1$ and $D_{max} = 6$, varying L/N in the range 1–40 produces a suitable range of hard problems; we will refer to these parameter settings as **PSa**. Figure 3 shows the median and worst-case solution times for **PSa** using all four decision procedures.

The formulae produced by our generator become unsatisfiable for much lower values of L/N and smaller formula size. For **PSa**, problems in the $\approx$50% satisfiable region are generated with values of L/N in the range 10–11. These are only one fifth the size of 50% satisfiable **PS1** formulae, and of similar difficulty (for DLP). Our generator also produces some hard unsatisfiable problems, although still in relatively small numbers, as well as some very hard satisfiable problems in the region where most formulae are trivially unsatisfiable (L/N in the range 30-40).

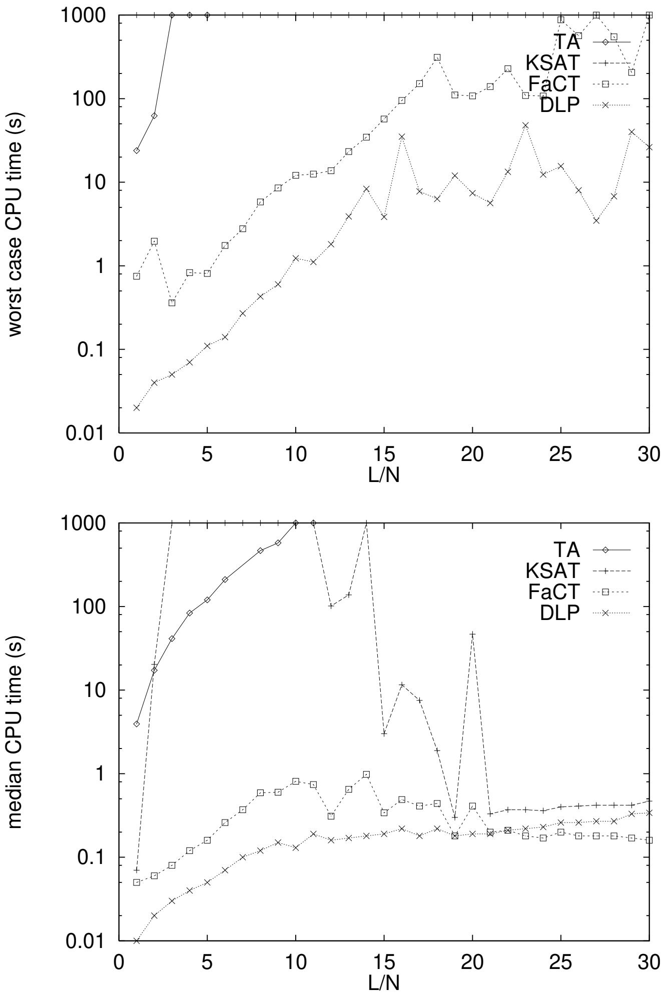

Figure 3: Worst-case and median CPU times for **PSa**

DLP tests

The differences in performance between the various systems is even more pronounced for **PSa** than was the case for **PS1**. KSAT in particular performs very badly, especially for satisfiable formulae: for values of L/N greater than 4 it was unable to solve any of the satisfiable problems within the 1,000s time limit. The performance of the FaCT system is interesting in that it shows some evidence of a easy-hard-easy pattern: median solution times reach their maximum in the ≈50% satisfiable region (L/N in the range 10–15) and subsequently diminish. Overall, DLP is the best performing system: its median solution time never exceeds 0.5s and it is able to solve all the problems within the 1,000s time limit.

In order to determine how the superior performance of DLP was related to its various optimisations,[1] the **PSa** test was repeated a number of times for configurations of

[1]For full details on these optimisations see [Horrocks and Patel-Schneider, 1998b].

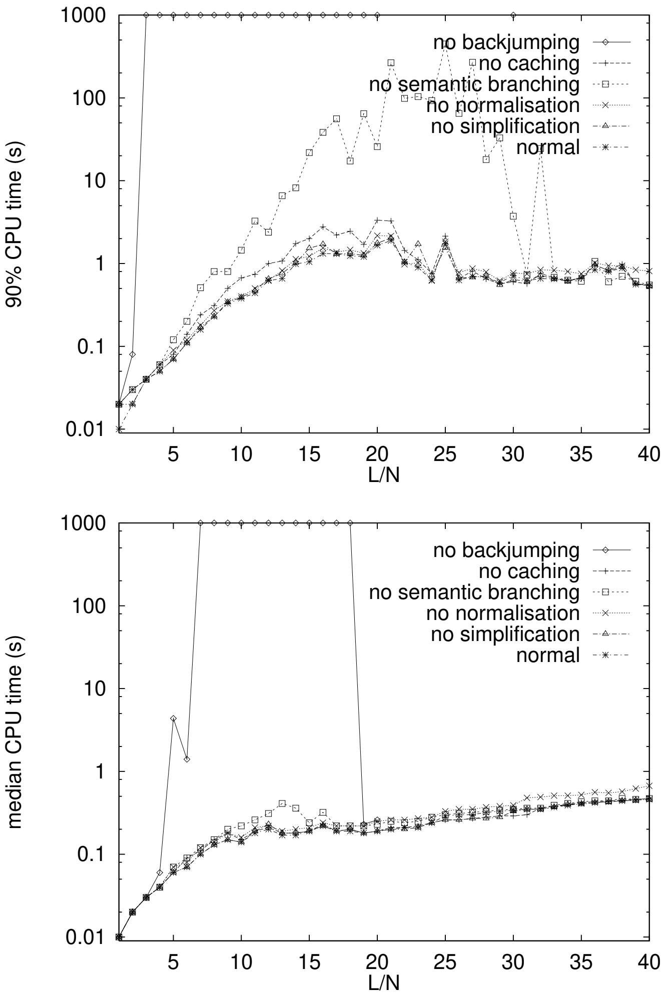

Figure 4: 90% and median CPU times for **PSa** with DLP

DLP in which one of the optimisations was disabled. Figure 4 shows the median and 90% solution times for DLP with each of its backjumping, result caching, semantic branching search, input formula normalisation and local simplification optimisations disabled.

From this is is clear the the backjumping optimisation is the main reason for DLP's performance advantage over KSAT and TA. This optimisation uses a form of dependency directed backtracking to avoid wasted search when contradictions result from early branching choices. In DLP, this optimisation is effective even when a contradiction derives from multiple modal nodes. Without backjumping, DLP performs very badly as L/N increases, although performance may recover somewhat for higher values of L/N; testing had to be abandoned in this area.

Semantic branching is the next-most-important optimisation, particularly for the harder problems. Caching also provides some benefit, although much less. Normalisation of the input formulae also provides some benefit, showing that there still remain some local tautologies or contradictions in the input.

There is some evidence of a easy-hard-easy pattern, most noticeable in the harder problems, although this is very slight unless the better-performing optimisations are turned off. Partly masking this easy-hard-easy pattern is the increasing time taken just to input the larger formulae.

Discussion

How the parameters controlling the generation of random modal formulae affect the difficulty of determining their satisfiability is as yet incompletely understood. It is therefore essential that generators for such random formulae be carefully designed and rigorously tested. Failing to do so can easily lead to results that reflect the characteristics of the generator, and their interaction with different poof techniques, rather than the characteristics of the underlying problem and of the the modal decision procedure(s) being tested.

Our results show that, contrary to earlier suggestions, increasing modal depth can produce much harder problems, particularly if the structure of the formulae generated is less uniform. They also demonstrate the importance of a spread of hard satisfiable and unsatisfiable problems as the performance of some systems may be far from uniform with respect to different problem types. It was also shown that DLP's dependency directed backtracking optimisation was the most important factor contributing to its superior performance. There was no conclusive evidence as to whether or not a phase shift can be observed for this type of problem: there was some positive indication of hard-easy-hard behaviour with both FaCT and DLP but not enough to be called a phase shift.

Another observation from the tests is that a small number of very hard "outlier" problems were generated. It is important that these problems are not masked by the analysis technique, for example by considering only median solution times, as the time taken to solve such problems could be critical to the utility of a decision procedure in realistic applications.

Although our generator is a considerable improvement, we are not completely satisfied with its performance. In particular, it still generates too few hard unsatisfiable problems and can still produce trivially unsatisfiable formulae. Moreover, it only generates $\mathbf{K_{(m)}}$ formulae, while current decision procedures are already able to deal with much more expressive logics including features such as transitive modalities and graded modalities.

Future work will therefore include further improvements to the generator in order to reduce the likelihood of trivially unsatisfiable formulae, extending the technique to generate formulae for more expressive logics, and more extensive testing and evaluation.

References

[Balsiger and Heuerding, 1998] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics — introduction and summary. In *Automated Reasoning with Analytic Tableaux and Related Methods: Int. Conf. Tableaux'98*, number 1397 in LNAI, pages 25–26. Springer-Verlag, 1998.

[Calvanese *et al.*, 1998] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proc. of the 9th Int. Workshop on Database and Expert Systems Applications (DEXA-98)*, pages 192–197, 1998.

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

[Giunchiglia and Sebastiani, 1996a] 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 Conf. on Automated Deduction (CADE-96)*, pages 583–597, 1996.

[Giunchiglia and Sebastiani, 1996b] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 304–314, 1996.

[Giunchiglia *et al.*, 1998] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 626–635, 1998.

[Heuerding and Schwendimann, 1996] 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.

[Hogg *et al.*, 1996] T. Hogg, B. A. Huberman, and C. P. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996. Editorial.

[Horrocks and Patel-Schneider, 1998a] I. Horrocks and P. F. Patel-Schneider. DL systems comparison. In *Collected Papers from the Int. Description Logics Workshop (DL'98)*, pages 55–57, 1998.

[Horrocks and Patel-Schneider, 1998b] I. Horrocks and P. F. Patel-Schneider. Optimising propositional modal satisfiability for description logic subsumption. In *Artificial Intelligence and Symbolic Computation: Int. Conf. AISC'98*, number 1476 in LNAI, pages 234–246. Springer-Verlag, 1998.

[Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.

[Hustadt and Schmidt, 1997a] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. Technical Report MPI-I-97-2-003, Max-Planck-Institut Für Informatik, 1997.

[Hustadt and Schmidt, 1997b] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)*, volume 1, pages 202–207, 1997.

[Patel-Schneider, 1998] P. F. Patel-Schneider. DLP system description. In *Collected Papers from the Int. Description Logics Workshop (DL'98)*, pages 87–89, 1998.

[Rector *et al.*, 1994] A. L. Rector, A. Gangemi, E. Galeazzi, A. J. Glowinski, and A Rossi-Mori. The GALEN core model schemata for anatomy: towards a re-useable application-independent model of medical concepts. In *Proc. of Medical Informatics in Europe*, pages 229–233, 1994.

[Selman *et al.*, 1996] B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.