An empirical analysis of modal theorem provers

Ullrich Hustadt*
Department of Computing and Mathematics,
Manchester Metropolitan University,
Manchester M1 5GD, United Kingdom

Renate A. Schmidt†
Department of Computing and Mathematics,
Manchester Metropolitan University,
Manchester M1 5GD, United Kingdom

ABSTRACT. This paper reports on an empirical performance analysis of four modal theorem provers on benchmark suites of randomly generated formulae. The theorem provers tested are the Davis-Putnam-based procedure KSAT*, the tableaux-based system $\mathcal{KRIS}$, the sequent-based Logics Workbench, and a translation approach combined with the first-order theorem prover SPASS.*

Our benchmark suites are sets of multi-modal formulae in a certain normal form randomly generated according to the scheme of Giunchiglia and Sebastiani [GS 96a, GS 96b]. We investigate the quality of the random modal formulae and show that the scheme has some shortcomings, which may lead to mistaken conclusions. We propose improvements to the evaluation method and show that the translation approach provides a viable alternative to the other approaches.
KEYWORDS: Modal logic, automated theorem proving, performance analysis.

Introduction

There are a variety of automated reasoning approaches for the basic propositional multi-modal logic $K(m)$ and its syntactical variant, the knowledge representation formalism $\mathcal{ALC}$ [SSS 91]. Some approaches utilize standard first-order theorem proving techniques in combination with translations from propositional modal logic to first-order logic [Ohl 91, Ohl 93, OS 97]. Others use Gentzen systems [Gob 74, HJSS 96]. Still others use tableaux proof methods [Fit 83, Rau 83, BH 91].

Usually, the literature on theorem provers for modal logic confines itself to a description of the underlying calculus and methodology accompanied with a

*Partially supported by the DFG grant Ga 261/8-1.
†Supported by the TRALOS-Project funded by the DFG.

consideration of the worst-case complexity of the resulting algorithm. Sometimes a small collection of benchmarks is given as in [Cat 91]. There have not been any exhaustive empirical evaluations or comparisons of the computational behaviour of theorem provers based on different methodologies.

Giunchiglia and Sebastiani [GS 96a, GS 96b] changed that. They report on an empirical analysis of the tableaux system $\mathcal{KRIS}$ [BH 91] and a new theorem prover, called KSAT. KSAT is an adaptation for the basic multi-modal logic $K(m)$ of a SAT-procedure for checking satisfiability in propositional logic. The benchmark suite is a set of randomly generated multi-modal formulae in a certain normal form.

We extend the empirical analysis of decision procedures for basic modal logic based on different methodologies by incorporating the Logics Workbench, a system based on a Gentzen-style calculus for modal logic [HJSS 96] and the functional translation approach of Ohlbach [Ohl 91]. The latter approach manipulates first-order translations of modal formulae, whereas the other three systems manipulate modal formulae directly. The four systems cover four different calculi and as far as we know, they are the only automated reasoners for modal logic that are publicly available.

Our investigations show benchmarking needs to be done with great care. The evaluation of Giunchiglia and Sebastiani has some shortcomings which we address. The random generator used to set up a benchmark suite produces formulae containing a substantial amount of tautologous and contradictory subformulae. It favours the SAT-procedure KSAT which utilizes a preprocessing routine that eliminates trivial tautologies and contradictions from the formulae. This property of the random formulae distorts the analysis and comparison of KSAT and $\mathcal{KRIS}$. The random generator does not produce challenging unsatisfiable modal formulae. So as to obtain harder problems we develop guidelines for the random generation of modal formulae. We present a set of samples of modal formulae generated according to these guidelines and verify that they provide challenging problems for KSAT and the translation approach.

The paper is structured as follows. Sections 1, 2, 3 and 4 briefly describe the inference mechanisms of KSAT, $\mathcal{KRIS}$, the Logics Workbench and the translation approach. Section 5 defines random modal formulae and describes the test method of Giunchiglia and Sebastiani, which Section 6 evaluates. Section 7 presents percentile graphs for the four systems that are more informative than graphs presenting the median CPU time consumption. Finally, Section 8 proposes improvements to the random generator so as to produce more challenging random samples on which the methods are tested. This paper is an extension of [HS 97].

1 The SAT-based procedure Ksat

The language of the multi-modal logic $K(m)$ is that of propositional logic plus m additional modal operators $\Box_i$. By definition, a *formula* of $K(m)$ is a boolean combination of propositional and modal atoms. A *modal atom* is an expression

of the form $\Box_i\psi$, where i is such that $1{\leq}i{\leq}m$ and ψ is a formula of $K(m)$. $\Diamond_i\psi$ is an abbreviation for $\neg\Box_i\neg\psi$. The semantics of $K(m)$ is given by the usual Kripke semantics.

KSAT tests the satisfiability of a given formula ϕ of $K(m)$. Its basic algorithm, called KSAT0, is based on the following two procedures:

KDP: Given a modal formula ϕ, this procedure generates a partial truth assignment μ for the propositional and modal atoms in ϕ which renders ϕ true propositionally. This is done using a decision procedure for propositional logic.

KM: Given a modal formula ϕ and an assignment μ computed by KDP, let $\Box_i\psi_{ij}$ denote any modal atom in ϕ which is assigned false by μ, that is, $\mu(\Box_i\psi_{ij}) = \bot$ and $\Box_i\phi_{ik}$ any modal atom that is assigned true by μ, that is, $\mu(\Box_i\phi_{ik}) = \top$. The procedure checks for each index i, $1{\leq}i{\leq}m$, and each j whether the formula

$$\varphi_{ij} = \bigwedge_k \phi_{ik} \wedge \neg\psi_{ij}$$

is satisfiable. This is done with KDP. If at least one of the formulae φ_{ij} is not satisfiable, then KM *fails on* μ, otherwise it *succeeds*.

KSAT0 starts by generating a truth assignment μ for ϕ using KDP. If KM succeeds on μ, then ϕ is $K(m)$-satisfiable. If KM fails on μ, we have to generate a new truth assignment for ϕ using KDP. If no further truth assignment is found, then ϕ is $K(m)$-unsatisfiable.

The decision procedure KDP for propositional logic can be described by a set of transition rules on ordered pairs $P \triangleright S$ where P is a sequence of pairs $\langle\phi, \mu\rangle$ of a modal formula ϕ and a partial truth assignment μ, and S is a set of satisfying truth assignments.

`dp_sol:` $$\frac{\langle\top, \mu\rangle \,|\, P \triangleright S}{P \triangleright S \cup \{\mu\}}$$

`dp_clash:` $$\frac{\langle\bot, \mu\rangle \,|\, P \triangleright S}{P \triangleright S}$$

`dp_unit:` $$\frac{\langle\phi[c], \mu\rangle \,|\, P \triangleright S}{\langle\phi', \mu \cup \{c = \top\}\rangle \,|\, P \triangleright S}$$

if c is a unit clause in ϕ and ϕ' is the result of replacing all occurrences of c and $\overline{c}$ by $\top$ and $\bot$, respectively, followed by boolean simplification.

`dp_split:` $$\frac{\langle\phi[m], \mu\rangle \,|\, P \triangleright S}{\langle\phi[m] \wedge p, \mu\rangle \,|\, \langle\phi[m] \wedge \neg p, \mu\rangle \,|\, P \triangleright S}$$

if `dp_unit` cannot be applied to $\langle\phi[m], \mu\rangle$, m is a propositional or modal atom.

The symbol | denotes concatenation of sequences.

Starting with $\langle \phi, \emptyset \rangle \triangleright \emptyset$, exhaustively applying the inference rules will result in $\emptyset \triangleright S$ where S is a complete set of partial truth assignments making ϕ true.

Note that the transition rules form a variant of the Davis-Putnam procedure for propositional formulae not in conjunctive normal form. The crucial nondeterminism of the procedure is the selection of the splitting 'variable' m in the transition rule `dp_split`. KSAT employs the heuristic that selects an atom with a maximal number of occurrences in ϕ.

At any point of time the computation in KDP can be interrupted and KM can be called with the partial truth assignment μ constructed so far. If KM fails on μ, then is not necessary to continue the completion of μ by KDP. KSAT0 calls KM before every application of the `dp_split` rule.

Giunchiglia and Sebastiani [GS 96a, pp. 583-584] suggest that KSAT0 can be based on any decision procedure for propositional logic. However, completeness of KSAT0 can easily be lost, even if the underlying propositional theorem prover is complete. Suppose that we add the pure literal rule to the Davis-Putnam procedure described above. That is, whenever an atom m occurs only positively (respectively negatively) in ϕ, we can add $\{m = \top\}$ (respectively $\{m = \bot\}$) to the truth assignment and replace all occurrences of m by $\top$ (respectively $\bot$). The application of the pure literal rule preserves satisfiability and can be applied eagerly to ϕ. Now consider the formula

$$\phi_1 = (p \vee q \vee \neg\Box_1(p \vee \neg p)) \wedge (\neg p \vee \neg q \vee \neg\Box_1(p \vee \neg p)).$$

There is one pure literal in ϕ_1, namely $\Box_1(p \vee \neg p)$, which occurs only negatively in ϕ_1. So we assign $\bot$ to $\Box_1(p \vee \neg p)$ and replace all occurrences of $\Box_1(p \vee \neg p)$ by $\bot$. After simplifying the resulting formula we get the formula $\top$. We have arrived at a truth assignment rendering ϕ true. Due to the eager application of the pure literal rule, this is the only truth assignment our procedure computes. In a second step we have to check using KM that $\neg(p \vee \neg p)$ is satisfiable. This is obviously not the case. Since KDP with the pure literal rule does not produce any additional truth assignments for ϕ, KSAT concludes that ϕ is unsatisfiable. However, ϕ is satisfiable with the truth assignment $\{p = \top, q = \bot\}$. So, legitimate optimizations of the decision procedure for propositional logic can render KSAT0 incomplete. That is, not every technique developed for such decision procedure carries over to modal logic.

We will illustrate the four satisfiability testing approaches under consideration by way of one satisfiable formula, namely

$$\psi = \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q).$$

Example 1

Figure 1 depicts the derivation tree of KSAT for the formula ψ. In the first step the procedure KDP applies the `dp_unit` rule to the unit clause $\neg\Box_1(p \vee r)$. All occurrences of $\neg\Box_1(p \vee r)$ are replaced by $\top$ while all occurrences of $\Box_1(p \vee r)$ are replaced by $\bot$. The resulting formula $\top \wedge (\Box_1 p \vee \Box_1 q)$ is simplified to $\Box_1 p \vee \Box_1 q$

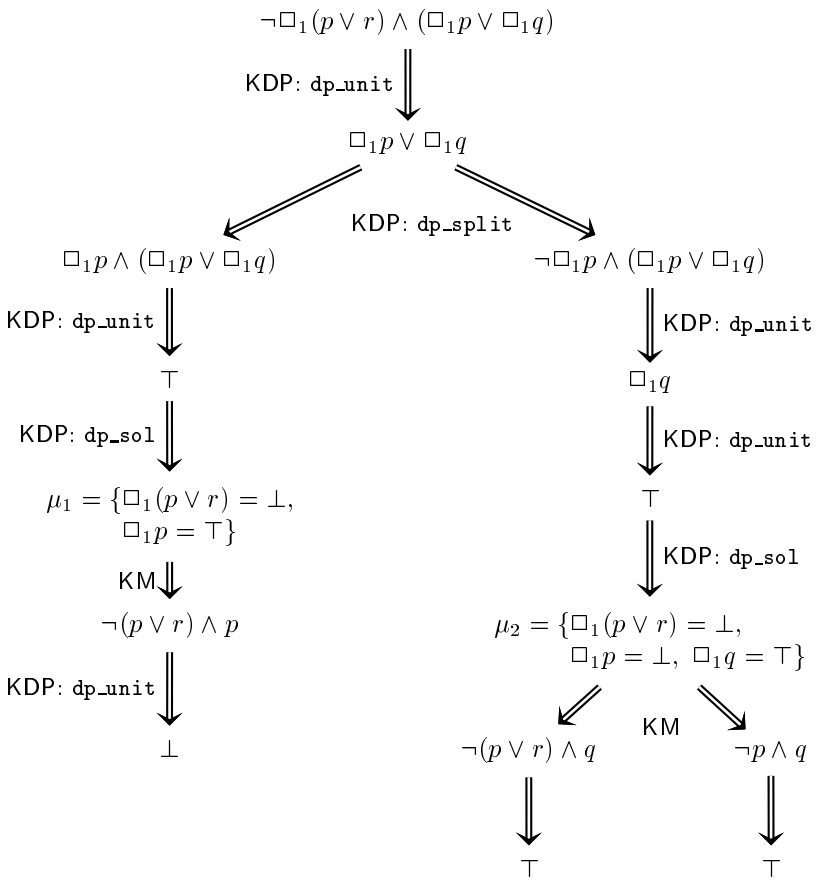


Figure 1: Sample derivation of KSAT

to which only the `dp_split` rule of KDP is applicable. Before any application of the `dp_split` rule, KSAT calls the procedure KM with the current truth assignment. Here, KM is used to prove that $\mu_0 = \{\Box_1(p \vee r) = \bot\}$ is $K(m)$-satisfiable. To this end, KM shows that $\neg(p \vee r)$ is satisfiable. This is done by KDP with two applications of the `dp_unit` rule to $\neg(p \vee r)$. Only now, the `dp_split` rule is actually applied to $\Box_1 p \vee \Box_1 q$. We assume that $\Box_1 p$ is the split variable. So, we have to show that either $\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ or $\neg\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ is satisfiable. KDP will first consider the formula $\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$. Obviously, we can apply the `dp_unit` rule to propagate the unit clause $\Box_1 p$. This step immediately reveals that the formula is satisfiable. That is, one satisfying truth assignment is $\mu_1 = \{\Box_1(p \vee r) = \bot, \Box_1 p = \top\}$. KSAT proceeds with KM to show that $\neg\Box_1(p \vee r) \wedge \Box_1 p$ is $K(m)$-satisfiable. This is done by showing that $\neg(p \vee r) \wedge p$ is satisfiable. But KDP will reveal with an application of the `dp_unit` rule to the unit clause p in $\neg(p \vee r) \wedge p$ that the formula is unsatisfiable. Thus, $\neg\Box_1(p \vee r) \wedge \Box_1 p$ is not $K(m)$-satisfiable. Consequently, KDP will continue with the second formula $\neg\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ generated by the `dp_split` rule. Here two applications of the `dp_unit` rule to the unit clauses $\neg\Box_1 p$ and $\Box_1 q$ yield a second truth assignment $\mu_2 = \{\Box_1(p \vee r) = \bot, \Box_1 p = \bot, \Box_1 q = \top\}$. Again KSAT continues with KM. Note that μ_2 assigns $\bot$ to two modal atoms, namely $\Box_1(p \vee r)$ and $\Box_1 p$. Therefore, KM checks the satisfiability of two propositional formulae, that is, $\neg(p \vee r) \wedge q$ and $\neg p \wedge q$. For both formulae KDP immediately verifies their satisfiability. So, KM succeeds on μ_2 which completes the computation by KSAT. We conclude that $\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$ is satisfiable.

2 The tableaux-based system $\mathcal{KRIS}$

While KSAT abstracts from the modal part of formulae to employ decision procedures for propositional logic, $\mathcal{KRIS}$ manipulates modal formulae directly. More precisely, the inference rules of $\mathcal{KRIS}$ are relations on sequences of sets of labelled modal formulae of the form $w{:}\psi$ where w is a label chosen from a countably infinite set of labels Γ and ψ is modal formula. For improved readability we write $w{:}\psi, C$ instead of $\{w{:}\psi\} \cup C$.

$$\bot\texttt{-elim:} \quad \frac{w{:}\bot, C \mid S}{S}$$

$$\top\texttt{-elim:} \quad \frac{w{:}\top, C \mid S}{C \mid S}$$

$$\wedge\texttt{-clash:} \quad \frac{w{:}\phi, w{:}\overline{\phi}, C \mid S}{S}$$

$$\wedge\texttt{-elim:} \quad \frac{w{:}\phi \wedge \psi, C \mid S}{w{:}\phi, w{:}\psi, C \mid S}$$

$\vee$-elim: $$\frac{w{:}\phi \vee \psi, C \mid S}{w{:}\phi, C \mid w{:}\psi, C \mid S}$$

if $w{:}\phi \vee \psi, C$ has been simplified by $\vee$-simp$_0$ and $\vee$-simp$_1$

$\Diamond_i$-elim: $$\frac{w{:}\Diamond_i\phi, D, C \mid S}{v{:}\phi \wedge \psi_1 \wedge \ldots \wedge \psi_n, D, C \mid S}$$

if $D = w{:}\Box_i\psi_1, \ldots, w{:}\Box_i\psi_n$, C does not contain any $w{:}\Box_i\psi$, none of the other rules can be applied to C, and v is a new label from Γ.

$\overline{\phi}$ denotes the complementary formula of ϕ, for example $\overline{\neg p} = p$ and $\overline{\Box_i p} = \Diamond_i \neg p$. Given a modal formula ϕ, the input sequence for $\mathcal{KRIS}$ is the singleton set $w_0{:}\phi'$, where w_0 is a label chosen from a countably infinite set of labels Γ and ϕ' is the modal negation normal form of ϕ. If $\mathcal{KRIS}$ arrives at a sequence $C \mid S$ such that no transformation rule can be applied to C, then the original formula ϕ is satisfiable. Otherwise the transformation rules will eventually reduce $w_0{:}\phi'$ to the empty sequence and ϕ is unsatisfiable. The rules $\bot$-elim, $\top$-elim, $\wedge$-clash, $\wedge$-elim are applied exhaustively before any application of one of the elimination rules for $\vee$ and $\Diamond_i$. The $\top$-elim rule is not necessary for the completeness of the set of rules.

In addition to the inference rules, $\mathcal{KRIS}$ has two simplification rules, namely

$\vee$-simp$_0$: $$w{:}\phi \vee \psi, w{:}\phi, C \rightarrow w{:}\phi, C$$

$\vee$-simp$_1$: $$w{:}\phi \vee \psi, w{:}\overline{\phi}, C \rightarrow w{:}\psi, w{:}\overline{\phi}, C$$

These are applied only immediately before an application of the $\vee$-elim rule and then they are applied only to the labelled formula $w{:}\phi \vee \psi$ to which we want to apply the $\vee$-elim rule.

As far as the application of the $\vee$-elim rule is concerned, $\mathcal{KRIS}$ actually considers the sets of labelled formulae as sequences and chooses the first disjunction in this sequence. To give a simple example, consider the formula ϕ_2 given by $(p \wedge \neg p) \vee \top$. Since ϕ_2 is in negation normal form, we start with the initial sequence

$$w_0{:}(p \wedge \neg p) \vee \top.$$

The only rule applicable is $\vee$-elim which generates the structure

$$w_0{:}(p \wedge \neg p) \mid w_0{:}\top.$$

For the reason that sequences are always processed from left to right, $w_0{:}(p \wedge \neg p)$ will be considered first. Only $\wedge$-elim is applicable transforming the sequence to

$$w_0{:}p, w_0{:}\neg p \mid w_0{:}\top.$$

Now we can apply the $\wedge$-clash rule to eliminate the first set of labelled formulae and get

$$w_0{:}\top.$$

A final application of the $\top$-`elim` rule reveals the sequence containing the empty set. No further rule can be applied. Since we have not arrived at the empty sequence, ϕ is satisfiable.

As the formula $(p \wedge \neg p) \vee \top$ is logically equivalent to $\top$, its satisfiability can be shown by a single application of the $\top$-elimination rule. However, $\mathcal{KRIS}$ has no simplification rules beside $\vee$-`simp`$_0$ and $\vee$-`simp`$_1$. In particular, $\mathcal{KRIS}$ does not simplify boolean expressions using the simplification rules of the pre-processing procedure that Giunchiglia and Sebastiani use in conjunction with KSAT which we discuss later (see Table 1 on page 18).

The condition that the $\Diamond_i$-`elim` rule can be applied only if none of the other rules can be applied to the set of labelled formulae under consideration is necessary for the completeness of the system. To illustrate the problem, consider the formula $\phi_3 = \neg q \wedge \Diamond_1 \neg p \wedge (\Box_1 p \vee q)$. Starting with

$$w_0{:}\neg q \wedge \Diamond_1 \neg p \wedge (\Box_1 p \vee q)$$

a sequence of applications of the $\wedge$-elimination rule will derive

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \vee q.$$

Suppose we apply the $\Diamond_1$-elimination rule before eliminating the occurrence of the $\vee$-operator in $w_0{:}\Box_1 p \vee q$. The resulting system is

$$w_0{:}\neg q, w_1{:}\neg p, w_0{:}\Box_1 p \vee q.$$

The application of $\vee$-elimination rule is still possible and we get

$$w_0{:}\neg q, w_1{:}\neg p, w_0{:}\Box_1 p \mid w_0{:}\neg q, w_1{:}\neg p, w_0{:}q.$$

Now, no further application of any inference rule is possible. Since, we have not derived the empty sequence, we would conclude that ϕ_3 is satisfiable. But, it is not. If we apply the $\vee$-elimination rule to

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \vee q$$

the resulting sequence contains two sets of labelled formulae

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \mid w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}q.$$

The only rule applicable to the first system is the $\Diamond_1$-elimination rule. The rule will replace the occurrence of $w_0{:}\Diamond_1 \neg p$ with $w_1{:}\neg p \wedge p$. We have now derived the sequence

$$w_0{:}\neg q, w_1{:}\neg p \wedge p, w_0{:}\Box_1 p \mid w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}q.$$

After an application of the $\wedge$-elimination rule we arrive at

$$w_0{:}\neg q, w_1{:}\neg p, w_1{:}p, w_0{:}\Box_1 p \mid w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}q.$$

It is straightforward to see that we can apply the $\wedge$-`clash` rule to both sets of labelled formulae. We end up with the empty sequence. Thus, ϕ_3 is unsatisfiable.

However, delaying the application of $\Diamond_1$-elimination to the end can also be a disadvantage. Consider, the structure

$$w_0{:}\Diamond_1\neg p, w_0{:}\Box_1 p, w_0{:}p \vee \Box_1 q.$$

Adding $w_1{:}\neg p \wedge p$ to the set of labelled formulae followed by an application of the $\wedge$-elimination and $\wedge$-clash rule allows the derivation of the empty sequence although we have not eliminated the disjunction in $p \vee \Box_1 q$ first. This test makes a difference computationally if the set of labelled formulae contains a large number of disjunctive formulae which are irrelevant with regards its satisfiability. It is possible to add the following $\Diamond_i$-`test` inference rule to the system without loosing completeness.

$\Diamond_i$-`test`: $$\frac{w{:}\Diamond_i\phi, D, C \mid S}{v{:}\phi \wedge \psi_1 \wedge \ldots \wedge \psi_n, w{:}\Diamond_i\phi, D, C \mid S}$$

if $D = w{:}\Box_i\psi_1, \ldots, w{:}\Box_i\psi_n$, and v is a new label chosen from Γ.

Furthermore, if we ensure that the rule is applied only finitely many times before we eventually eliminate $w{:}\Diamond_i\phi$ by the $\Diamond_i$-elimination rule, the inference system remains terminating. Note that the application of the $\Diamond_i$-`test` rule closely resembles the intermediate calls of the KM procedure during a computation of KDP by KSAT.

We end our description of the system $\mathcal{KRIS}$ with a sample derivation.

Example 2
Again, we consider the satisfiable formula $\psi = \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. First, it transforms the formula ψ to its negation normal form ψ' which is $\psi' = \Diamond_1(\neg p \wedge \neg r) \wedge (\Box_1 p \vee \Box_1 q)$. Figure 2 shows how $\mathcal{KRIS}$ proceeds to prove the satisfiability of ψ'. First, $\mathcal{KRIS}$ eliminates the occurrence of the $\wedge$-operator in ψ'. Then it uses the $\vee$-`elim` rule to split the disjunctive formula $(\Box_1 p \vee \Box_1 q)$. Now we have to deal with two sets of labelled formulae. $\mathcal{KRIS}$ continues with the left set $w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p$. The only rule applicable to this set is $\Diamond_1$-`elim`. The application of the $\Diamond_1$-`elim` rule eliminates the labelled formula $w_0{:}\Diamond_1(\neg p \wedge \neg r)$ from our set and adds $w_1{:}\neg p \wedge \neg r \wedge p$. Applying the $\wedge$-`elim` rule to this labelled formula reveals that our set of labelled formulae contains both $w_1{:}\neg p$ and $w_1{:}p$. This is a contradiction and the $\wedge$-`clash` rule eliminates this set of labelled formula from the sequence. The remaining set of labelled formulae, namely $w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q$, is the second set generated by the $\vee$-`elim` rule. Again, the only applicable rule is $\Diamond_1$-`elim`. This adds the formula $w_1{:}\neg p \wedge \neg r \wedge q$ to the set while removing $w_0{:}\Diamond_1(\neg p \wedge \neg r)$. A sequence of applications of the $\wedge$-`elim` rule results in a set of labelled formulae to which no further rule applies. Thus, $\mathcal{KRIS}$ has shown that ψ' and ψ are satisfiable.

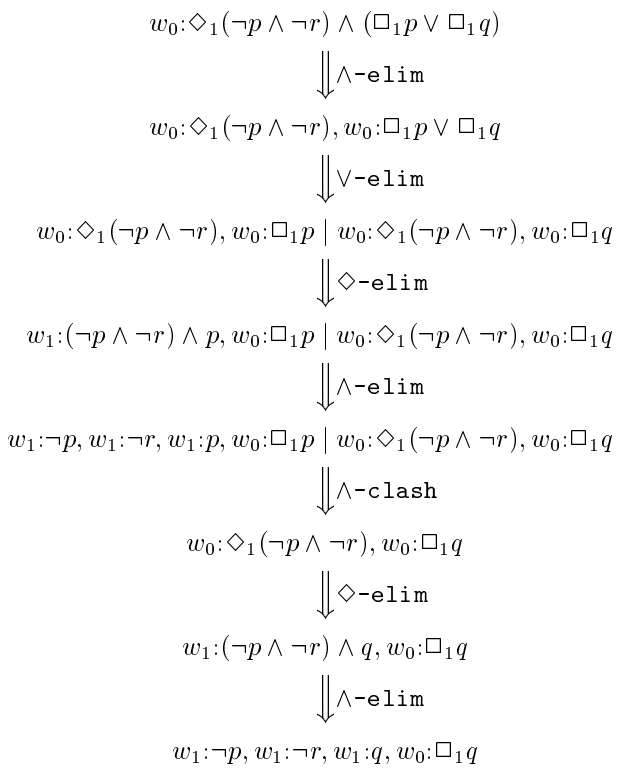


Figure 2: Sample derivation of $\mathcal{KRIS}$

Axioms:

$$\phi, \Gamma \Rightarrow \phi, \Delta \qquad \Gamma \Rightarrow \top, \Delta \qquad \bot, \Gamma \Rightarrow \Delta$$

Rules:

$$\frac{\phi, \psi, \Gamma \Rightarrow \Delta}{\phi \wedge \psi, \Gamma \Rightarrow \Delta}\ (l\wedge) \qquad \frac{\Gamma \Rightarrow \phi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}\ (r\wedge)$$

$$\frac{\phi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\phi \vee \psi, \Gamma \Rightarrow \Delta}\ (l\vee) \qquad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\ (r\vee)$$

$$\frac{\Gamma \Rightarrow \phi, \Delta}{\neg\phi, \Gamma \Rightarrow \Delta}\ (l\neg) \qquad \frac{\phi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}\ (r\neg)$$

$$\frac{\phi, \Gamma \Rightarrow \Delta}{\Diamond_i\phi, \Box_i\Gamma, \Sigma \Rightarrow \Diamond_i\Delta, \Pi}\ (l\Diamond_i) \qquad \frac{\Gamma \Rightarrow \phi, \Delta}{\Box_i\Gamma, \Sigma \Rightarrow \Box_i\phi, \Diamond_i\Delta, \Pi}\ (r\Box_i)$$

Figure 3: Axioms and rules of the Logics Workbench

3 The Logics Workbench

The Logics Workbench (LWB) is an interactive system providing inference mechanisms for a variety of logical formalisms including basic modal logic. The decision procedure for basic modal logic is based on the sequent calculus presented in Figure 3 [HJSS 96] (of which some axioms and rules are eliminable). A modal formula ϕ is derivable using the axioms and rules of the sequent calculus if and only if ϕ is true in all Kripke models. Since we are interested in satisfiability not provability, we exploit that a given formula ϕ is unsatisfiable if and only if $\neg\phi$ is provable using the calculus of the Logics Workbench.

Unlike $\mathcal{KRIS}$, the Logics Workbench has no simplification rules. For example, a sequent proof of the satisfiability of the formula $\neg p \wedge (p \vee q)$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \Rightarrow p \quad \dfrac{\text{Failure}}{q \Rightarrow p}}{(p \vee q) \Rightarrow p}\ (l\vee)}{\neg p, (p \vee q) \Rightarrow}\ (l\neg)}{\neg p \wedge (p \vee q) \Rightarrow}\ (l\wedge)}{\Rightarrow \neg(\neg p \wedge (p \vee q))}\ (r\neg)$$

Starting with the sequent $\Rightarrow \neg(\neg p \wedge (p \vee q))$, the Logics Workbench conducts a backwards proof search. That is, the inference rules presented in Figure 3 are applied bottom up. The $(r\neg)$-rule moves the formula $\neg p \wedge (p \vee q)$ to the left side of the sequent. Then we eliminate the occurrence of the conjunctive operator using the $(l\wedge)$-rule. The left hand side of the sequent now consist of two formulae, namely $\neg p$ and $(p \vee q)$. It uses the $(l\neg)$-rule to move $\neg p$ to the

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \Rightarrow p, r}{p \Rightarrow p \vee r}\,(r\vee)}{\Box_1 p \Rightarrow \Box_1(p \vee r)}\,(r\Box_1) \quad \dfrac{\dfrac{\text{Failure}}{q \Rightarrow p, r}\,(r\vee)}{q \Rightarrow p \vee r}\,(r\Box_1)}{\Box_1 q \Rightarrow \Box_1(p \vee r)}\,(l\vee)}{\Box_1 p \vee \Box_1 q \Rightarrow \Box_1(p \vee r)}\,(l\neg)}{\neg\Box_1(p \vee r), \Box_1 p \vee \Box_1 q \Rightarrow}\,(l\wedge)}{\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q) \Rightarrow}\,(r\neg)}{\Rightarrow \neg(\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q))}$$

Figure 4: Sample derivation of the Logics Workbench

right-hand side of the sequent. Now the $(l\vee)$-rule is the only rule applicable to the sequent $(p \vee q) \Rightarrow p$ we have arrived at. We get two sequents, namely $p \Rightarrow p$ and $q \Rightarrow p$. Only the first one is an axiom. The sequent $q \Rightarrow p$ is neither an axiom nor can we apply any further rules of the calculus. We have failed to construct a proof of $\Rightarrow \neg(\neg p \wedge (p \vee q))$. Therefore $\neg p \wedge (p \vee q)$ is satisfiable.

There are two points worth noting. An application of the $(l\vee)$-rule creates two branches into our backwards proof search. If one of the branches fails, the whole proof attempt fails. We could directly derive the sequent $\neg p, q \Rightarrow$ from $\neg p, (p \vee q) \Rightarrow$ using the equivalent of the $\vee$-$\mathtt{simp}_1$ rule for sequents. This would eliminate the need to apply the $(l\vee)$-rule in the example. But, as mentioned before, the Logics Workbench has no equivalents of the $\vee$-simplification rules.

However, the Logics Workbench uses the following form of branch pruning. Provided in a backwards application of the $(l\vee)$-rule the formula ϕ is not used in the proof of $\phi, \Gamma \Rightarrow \Delta$, that is, $\Gamma \Rightarrow \Delta$ holds, then it is not necessary to consider the branch $\psi, \Gamma \Rightarrow \Delta$. Similarly, branch pruning is applied to the $(r\wedge)$-rule.

The Logics Workbench applies the $(l\wedge)$-rule, $(l\neg)$-rule, $(r\neg)$-rule and $(r\neg)$-rule exhaustively before any application of the remaining rules. The selection of the disjunctive and conjunctive formulae for applications of the $(l\vee)$-rule and $(r\wedge)$-rule, respectively, is determined by the order of formulae in the left-hand side and right-hand side of the sequent, respectively. The $(l\Diamond_i)$-rule and $(r\Box_i)$-rule are applied only after no application of the other rules is possible.

Example 3
Figure 4 gives the derivation produced by the Logics Workbench of the satisfiability of $\psi = \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. Starting from $\Rightarrow \neg(\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q))$ the backwards applications of the $(r\vee)$-rule, $(l\wedge)$-rule and $(l\neg)$-rule lead to the sequent $\Box_1 p \vee \Box_1 q \Rightarrow \Box_1(p \vee r)$. The backwards application of the $(l\vee)$-rule generates two sequents $\Box_1 p \Rightarrow \Box_1(p \vee r)$ and $\Box_1 q \Rightarrow \Box_1(p \vee r)$. The Logics Workbench first considers the sequent $\Box_1 p \Rightarrow \Box_1(p \vee r)$. Here we have to apply the $(r\Box_1)$-rule, for which we have to select a formula of the form $\Box\phi$

on the right-hand side of the sequent. Since in the sequent under consideration only one $\Box$-formula occurs on the right-hand side of the sequent, the choice is deterministic. The application of the $(r\Box_1)$-rule yields the sequent $p \Rightarrow p \vee r$. With a final application of the $(r\vee)$-rule we arrive at the axiom $p \Rightarrow p, r$. Now the Logics Workbench turns to the second alternative $\Box_1 q \Rightarrow \Box_1(p \vee r)$. Here the application of the $(r\Box_1)$-rule produces $q \Rightarrow p \vee r$. An application of the $(r\vee)$-rule renders $q \Rightarrow p, r$. Since no more rules apply and $q \Rightarrow p, r$ is not an axiom, our attempt to construct a proof fails. No other proof attempts are possible. So ψ is satisfiable.

Observe the near correspondence between the proof search of $\mathcal{KRIS}$ and that of the Logics Workbench. We can directly translate the deduction steps in the tableaux-calculus of $\mathcal{KRIS}$ into the sequent calculus of the Logics Workbench. The differences are the absence of simplification rules in the Logics Workbench, the presence of branch pruning in the Logics Workbench, and the conversion to negation normal form by $\mathcal{KRIS}$.

4 The translation approach

The translation approach (TA) is based on the idea that modal inference can be done by translating modal formulae into first-order logic and conventional first-order theorem proving. The translation approach we use is the optimized functional translation approach described in Ohlbach and Schmidt [OS 97]. It has the property that ordinary resolution without any refinement strategies is a decision procedure for $K(m)$ [Sch 98b, Sch 98a]. The translation maps modal formulae into a logic, called basic path logic, which is a monadic fragment of sorted first-order logic with one binary function symbol $\circ$ that defines accessibility. A formula of path logic is further restricted in that its clausal form may only contain Skolem terms that are constants.

The optimized functional translation does a sequence of transformations. The first transformation Π_f maps a modal formula ϕ to its so-called functional translation defined by $\Pi_f(\phi) = \forall x\ \pi_f(\phi, x)$. For $K(m)$, π_f is defined by

$$\begin{array}{rcl} \pi_f(p, s) & = & P(s) \\ \pi_f(\Box_i \phi, s) & = & \mathit{def}_i(s) \rightarrow \forall \alpha_i\ \pi_f(\phi, s \circ \alpha_i). \end{array}$$

p is a propositional variable and P is a unary predicate uniquely associated with p. The symbol def_i is a special unary predicate with sort i that specifies definability (or not dead end), replacing $\neg \mathit{de}_i$ in the definition of Ohlbach and Schmidt. α_i denotes a variable of sort i. For the propositional connectives π_f is a homomorphism. The second transformation applies the so-called quantifier exchange operator Υ which moves existential quantifiers inwards over universal quantifiers using the rule '$\exists\alpha\forall\beta\,\psi$ becomes $\forall\beta\exists\alpha\,\psi$'. The transformation Υ preserves satisfiability, more specifically, ϕ is a theorem in $K(m)$ if and only if $\neg\Upsilon\Pi_f(\phi)$ is unsatisfiable [OS 97] (the quantifier exchange operator rests on the generated model property and the fact that generated models are trees).

Our aim is to test the satisfiability of a given modal formula ϕ. This can be achieved by testing the satisfiability of the set of clauses $S = c(\neg\Upsilon\Pi_f(\neg\phi))$, where $c(\psi)$ denotes the clausal form of a first-order formula ψ. S is a set of clauses in the basic path logic.

For $K(m)$ additional transformations of the clause set S are possible. First, we replace all occurrences of literals $P(s)$ where s is a path of the form $x \circ u^1_{i_1} \circ u^2_{i_2} \circ \cdots \circ u^n_{i_n}$ with length $n+1$ by $P_{n+1}(x, u^1_{i_1}, \ldots, u^n_{i_n})$ where P_{n+1} is an $(n+1)$-ary predicate symbol uniquely associated with P and n. This can be done since $\circ$ is not associative and admissible substitutions either rename variables or do instantiation with constants. Second, the sort information associated with the variables and constants occurring in the literals in the clause set can be encoded in the predicate symbols of the literals. So, we can replace all occurrences of literals $P_{n+1}(x, u^1_{i_1}, \ldots, u^n_{i_n})$ by $P_{i_1 \ldots i_n}(x, u^1, \ldots, u^n)$ where $P_{i_1 \ldots i_n}$ is a predicate symbol uniquely associated with the predicate symbol P_{n+1} and the sorts $i_1, \ldots, i_n$. The variables and constants $u^1, \ldots, u^n$ no longer carry any sort information. Finally, we observe that all literals in the transformed clause set share the first argument x, which we can eliminate safely. This sequence of three transformations can be combined into one:

$$P(x \circ u^1_{i_1} \circ u^2_{i_2} \circ \cdots \circ u^n_{i_n}) \quad \text{becomes} \quad P_{i_1 \ldots i_n}(u^1, \ldots, u^n).$$

Example 4
We consider our example formula ψ given by $\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. The result of $c(\neg\Upsilon\Pi_f(\neg\psi))$ is a set of four clauses, namely

(1) $\quad def_1$

(2) $\quad \neg P_1(\underline{a})$

(3) $\quad \neg R_1(\underline{a})$

(4) $\quad \neg def_1 \vee \neg def_1 \vee P_1(x) \vee Q_1(y)$

Two resolution steps are possible: Resolving clauses (1) and (4) yields $P_1(x) \vee Q_1(y)$. The derived clause subsumes the clause (4). Resolving $P_1(x) \vee Q_1(y)$ with clause (2) yields the unit clause $Q_1(y)$, that subsumes the clause $P_1(x) \vee Q_1(y)$. Subsumption leaves the following clause set on which no further inference steps are possible.

$$\begin{array}{l} def_1 \\ \neg P_1(\underline{a}) \\ \neg R_1(\underline{a}) \\ Q_1(y) \end{array}$$

Since the final clause set does not contain the empty clause, the original clause set, and consequently, the modal formula ϕ is satisfiable.

For theorem proving we use FLOTTER and SPASS Version 0.55 [Wei 97]. FLOTTER is a system that computes the clausal normal form of a given first-order formula. It performs the following steps.

1. Rename subformulae of the input formula in order to obtain a clause set containing a minimal number of clauses. Here an improved variant of the technique developed by Boy de la Tour [BdlT 92] is used.

2. Remove implications and equivalences using the appropriate transformation rules.

3. Compute the negation normal form.

4. Eliminate existential quantifiers by Skolemization.

5. Compute the clausal normal form.

6. Test the resulting clause set for redundancy by subsumption, tautology removal and condensing.

The theorem prover SPASS is based on the superposition calculus of Bachmair and Ganzinger [BG 90] extended with the sort techniques of Weidenbach [Wei 96].

We opted to use SPASS and not other well-known theorem provers (like OTTER) for the following reasons:

1. SPASS uses ordered resolution and ordered factoring based on an extended Knuth-Bendix ordering [Pet 83].

2. It supports splitting and branch condensing. Splitting amounts to case analysis while branch condensing resembles branch pruning in the Logics Workbench.

3. It has an elaborated set of reduction rules including tautology deletion, subsumption, and condensing.

Ordered inference rules and splitting are of particular importance when treating satisfiable formulae. Also, SPASS supports dynamic sort theories by additional inference rules including sort generation and sort resolution and additional reduction rules like sort simplification and clause deletion. It considers every unary predicate symbol as a sort (not to be confused with the sorts of the translation morphism). The translation of random 3CNF formulae will result in first-order formulae which contain a great number of such symbols.

5 The evaluation method

The evaluation method adopted by Giunchiglia and Sebastiani follows the approach of Mitchell, Selman, and Levesque [MSL 92]. To set up a benchmark suite for Davis-Putnam-based theorem provers Mitchell et al. [MSL 92] generate propositional formulae using the fixed clause-length model. Giunchiglia and Sebastiani modify this approach for the modal logic $K(m)$.

There are five parameters: the number of propositional variables N, the number of modalities M, the number of modal subformulae per disjunction K,

the number of modal subformulae per conjunction L, the modal degree D, and the probability P. Based on a given choice of parameters random modal KCNF formulae are defined inductively as follows. A *random (modal) atom* of degree 0 is a variable randomly chosen from the set of N propositional variables. A *random modal atom* of degree D, $D>0$, is with probability P a random modal atom of degree 0 or an expression of the form $\Box_i\phi$, otherwise, where $\Box_i$ is a modality randomly chosen from the set of M modalities and ϕ is a random modal KCNF clause of modal degree $D-1$ (defined below). A *random modal literal* (of degree D) is with probability 0.5 a random modal atom (of degree D) or its negation, otherwise. A *random modal KCNF clause* (of degree D) is a disjunction of K random modal literals (of degree D). Now, a *random modal KCNF formula* (of degree D) is a conjunction of L random modal KCNF clauses (of degree D).

For the comparison of the performance of KSAT and $\mathcal{KRIS}$, Giunchiglia and Sebastiani proceed as follows. All parameters are fixed except L, the number of clauses in a formula. For example, N=3, M=1, K=3, D=5, and P=0.5 are chosen. The parameter L ranges from N to $40N$. For each value of the ratio L/N a set of 100 random modal KCNF formulae of degree D is generated. We will see that for small L the generated formulae are most likely to be satisfiable and for larger L the generated formulae are most likely to be unsatisfiable. For each generated formula ϕ, the time needed by one of the decision procedures to determine the satisfiability of ϕ is measured. This includes also the time consumed by preprocessing steps like simplification and translation of the input formulae. There is an upper limit for the CPU time consumed, more precisely, the amount of time during which computational effort has been expended on behalf of the executing procedure. As soon as the upper limit is reached, the computation for ϕ is stopped. Now, the median CPU runtime over the ratio L/N is presented. For example, the graphs of Figure 5 show the performance of $\mathcal{KRIS}$ and KSAT on the parameter settings **PS0** (N=5, M=1, K=3, D=2, P=0.5) and **PS1** (N=3, M=1, K=3, D=5, P=0.5). The gaps in the graphs (for example for $\mathcal{KRIS}$ above $L/N = 5$) indicate that the computations on more than 50 out of 100 formulae of given ratio L/N had to be abandoned.

Our tests were run on a Sun Ultra 1/170E with 196MB main memory running SunOS 5.5.1 using a time-limit of 1000 CPU seconds. For KSAT and $\mathcal{KRIS}$ which are both written in Common Lisp we have used versions of the recommended compilers: for KSAT `AKCL` (Austin Kyoto Common Lisp) version 1.625, and for $\mathcal{KRIS}$ `CLISP` version 1996-03-15. For the Logics Workbench which is written in C++ we have used a pre-compiled executable of version 1.0 obtained from `http://lwbwww.unibe.ch:8080/`. We used SPASS and FLOTTER version 0.55, written in ANSI C and compiled using the GNU C compiler version 2.7.1.

Giunchiglia and Sebastiani [GS 96b] present graphs for the following pa-

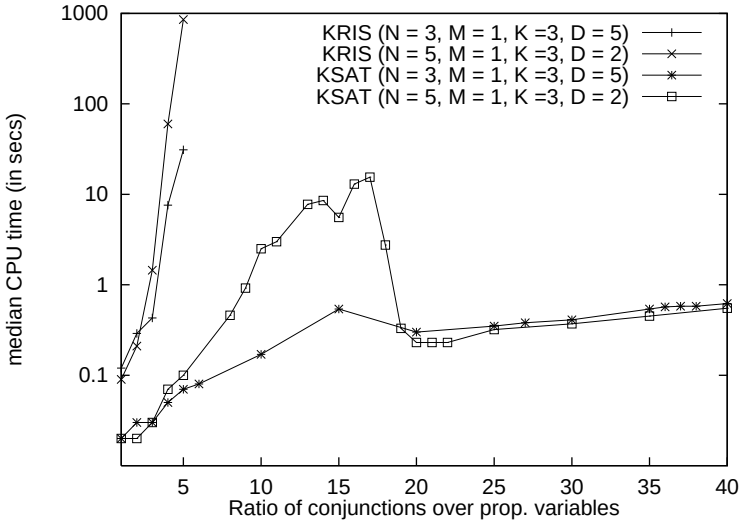


Figure 5: The performance of $\mathcal{KRIS}$ and KSAT for **PS0** and **PS1**

rameter settings:

	N	M	K	D	P		N	M	K	D	P
PS0	5	1	3	2	0.5	**PS5**	4	1	3	2	0.5
PS1	3	1	3	5	0.5	**PS6**	4	2	3	2	0.5
PS2	3	1	3	4	0.5	**PS7**	4	5	3	2	0.5
PS3	3	1	3	3	0.5	**PS8**	4	10	3	2	0.5
PS4	3	1	3	2	0.5	**PS9**	4	20	3	2	0.5

Based on their graphs they come to the following conclusions [GS 96b, p. 313]:

(1) KSAT outperforms by orders of magnitude the previous state-of-the art decision procedures.

(2) All SAT-based modal decision procedures are intrinsically bound to be more efficient than tableaux-based decision procedures.

(3) There is partial evidence of an easy-hard-easy pattern on randomly generated modal logic formulae independent of all the parameters of evaluation considered.

The graphs for the parameter settings **PS0** and **PS1** of Figure 5 support these claims most visibly. We show that the situation is more complex and does not justify such strong claims. We focus on the parameter settings **PS0** and **PS1** which suffice for our analysis of the evaluation method in the next section. The remaining parameter settings will be considered in Section 8.

6 Analysis of the evaluation method

Selecting good test instances is crucial when evaluating and comparing the performances of algorithms empirically. This means we have to determine the characteristics of the test instances before starting a performance comparison. This is particularly important when we set up a completely new collection of test instances. We address the question which characteristics the formulae produced by the random generator and the parameter settings chosen by Giunchiglia and Sebastiani have and how they influence the theorem provers under consideration with respect to claims (1) to (3).

It is important to note that for $D{=}0$ and $K{=}3$ random modal KCNF formulae do *not* coincide with random 3SAT formulae. Generating a clause of a random 3SAT formula means randomly generating a *set* of three propositional variables and then negating each member of the set with probability 0.5. In contrast, generating a random modal 3CNF clause of degree 0 means randomly generating a *multiset* of three propositional variables and negating each member of the multiset with probability 0.5. For example, $p \vee q \vee \neg r$ is a 3SAT clause and also a modal 3CNF clause of degree 0. The clauses $p \vee \neg p \vee p$ and $p \vee p \vee q$ are not random 3SAT clauses, but both are random modal 3CNF clauses of degree 0. In random modal 3CNF formulae of higher degree, such clauses occur within the scope of a modal operator. For example, contradictory expressions like $\neg\Box_1(p \vee \neg p \vee p)$ may occur. Thus, random modal KCNF formulae contain tautological and contradictory subformulae. It is easy to remove these subformulae without affecting satisfiability. We now consider to what extent the size of the random modal 3CNF formulae can be reduced by such a simplification. The graphs of Figure 6 reflect the average ratio of the size of the simplified random modal 3CNF formulae over the size of the original formulae. For the random modal 3CNF formulae generated using three propositional variables, on average, the size of a simplified formula is only $1/4$ of the size of the original formula. For the second parameter setting we observe a reduction to $1/2$ of the original size. In other words, one half to three quarters of the random modal 3CNF formulae is "logical garbage" that can be eliminated at little cost.

KSAT utilizes a form of preprocessing that removes duplicate and contradictory subformulae of an input formula, by applying the simplification rules presented in Table 1. The rules simplify $p \vee q \vee p$ to $p \vee q$ and $\Box_1(p \vee q) \wedge \neg\Box_1(p \vee q)$ to $\bot$, but they will not simplify $\Box_1(p \vee q) \wedge \neg\Box_1(q \vee p)$ to $\top$, since $\Box_1(p \vee q)$ is not

$\neg\phi \vee \phi \to \top$	$\phi \vee \top \to \top$	$\phi \vee \bot \to \phi$	$\phi \vee \phi \to \phi$
$\neg\phi \wedge \phi \to \bot$	$\phi \wedge \top \to \phi$	$\phi \wedge \bot \to \bot$	$\phi \wedge \phi \to \phi$
$\neg\bot \to \top$	$\neg\top \to \bot$	$\Box_i\top \to \top$	

Table 1: The simplification rules of KSAT

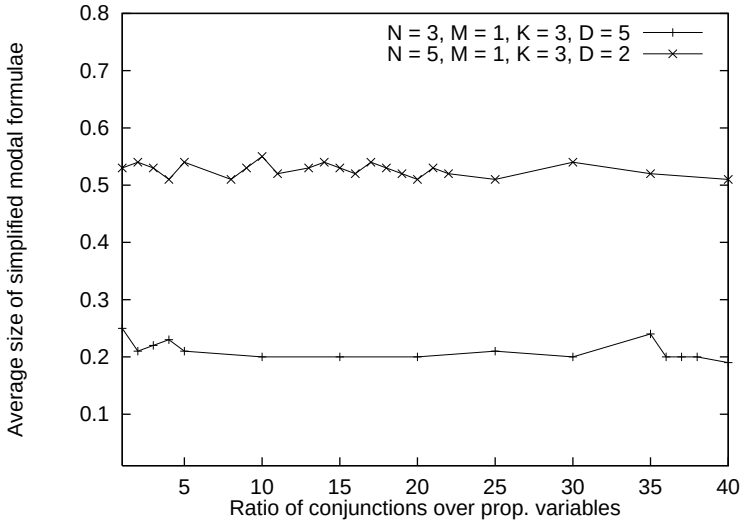


Figure 6: The effect of simplifying modal 3CNF formulae

syntactically equal to $\Box_1(q \vee p)$. KSAT also sorts disjunctions lexicographically, e.g. $\Box_1(q \vee p)$ will be replaced by $\Box_1(p \vee q)$. This allows for additional applications of the simplification rules. However, in all our experiments we have chosen to disable the reordering inside KSAT. For the median CPU runtime considerations of this section, reordering has no significant effect as Figure 7 shows. Likewise the other approaches take no advantage of reordering as performed in KSAT. But we think reordering is an important notion that deserves further investigation, for all procedures. In particular, generalizing the notion of reordering as implemented inside KSAT to a notion of reordering of conjunctions of clauses, will have a positive effect on the Logics Workbench and $\mathcal{KRIS}$.

KSAT performs exactly the simplification whose effect is illustrated in Figure 6. $\mathcal{KRIS}$, on the other hand, does not. As no theorem prover except for KSAT is designed especially for formulae with characteristics similar to those of the random modal formulae, we think it is fair that either no theorem prover should use preprocessing or all should. Simplification of the generated modal formulae is reasonable, so we have added the preprocessing function of KSAT also to the other theorem provers that we consider. The modified versions of $\mathcal{KRIS}$, the Logics Workbench and the translation approach with preprocessing will be denoted by $\mathcal{KRIS}$*, LWB* and TA*, respectively. The graphs in Figure 8 show the performances of KSAT and $\mathcal{KRIS}$*. Although the performance of KSAT is still better than that of $\mathcal{KRIS}$*, the picture is completely different than that of Figure 5.

The superior performance of KSAT diminishes if we turn to values of the

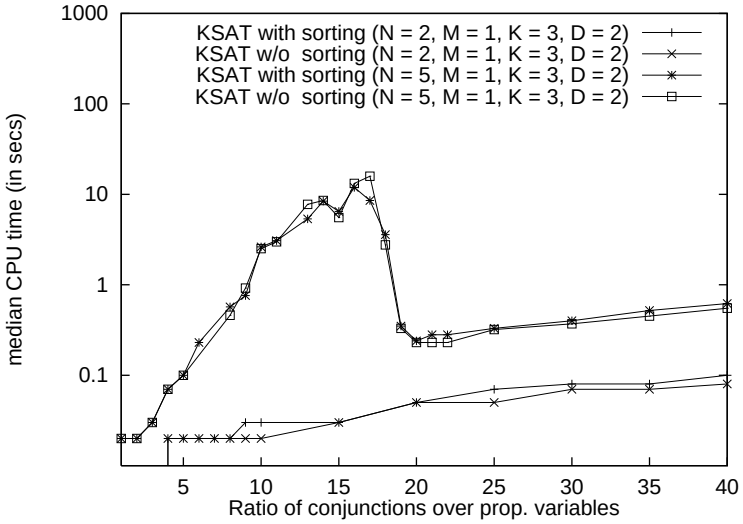


Figure 7: The performance of KSAT with/without reordering

parameter N greater than 5. Figure 9 shows the performance of KSAT and $\mathcal{KRIS}$* on the parameter setting **PS10** (N=8, M=1, K=3, D=2, P=0.5), while Figure 10 shows the performance on the parameter setting **PS11** (N=10, M=1, K=3, D=2, P=0.5). We see that the performance of $\mathcal{KRIS}$* for a ratio L/N between 4 and 11 on **PS10** and for a ratio L/N between 3 and 9 on **PS11** is better than the performance of KSAT. So, it is not true that KSAT outperforms $\mathcal{KRIS}$* in general, which relativizes claim (1). For increased numbers of propositional variables, the `dp_unit` rule and exhaustive boolean simplification of KSAT is of no particular importance for modal formulae which are likely satisfiable. And, the intermediate calls to KM before each application of the `dp_split` have a deteriorating effect on the performance.

$\mathcal{KRIS}$* applies the $\vee$-elimination rule to every disjunction in the modal formula and continues on the first branch. As the number of propositional variables and modal atoms is large, the $\wedge$`-clash` rule is less likely to close a branch and the second branch need not be treated. After all occurrences of the $\vee$-operator are eliminated, $\mathcal{KRIS}$* performs all possible applications of the $\Diamond_i$`-elim` rule. Each application is likely to succeed.

By contrast, KSAT uses `dp_split` to generate two possible extensions of the current truth assignment. Like $\mathcal{KRIS}$*, it rarely has to consider the second extension at all. However, before every application of the `dp_split` rule the procedure KM is called. This has the following effect: The `dp_split` rule needs to be applied more often before reaching a satisfying truth assignment, since the number of different propositional variables and modal atoms has become larger. This also holds for the recursive calls of KDP by KM. There is an

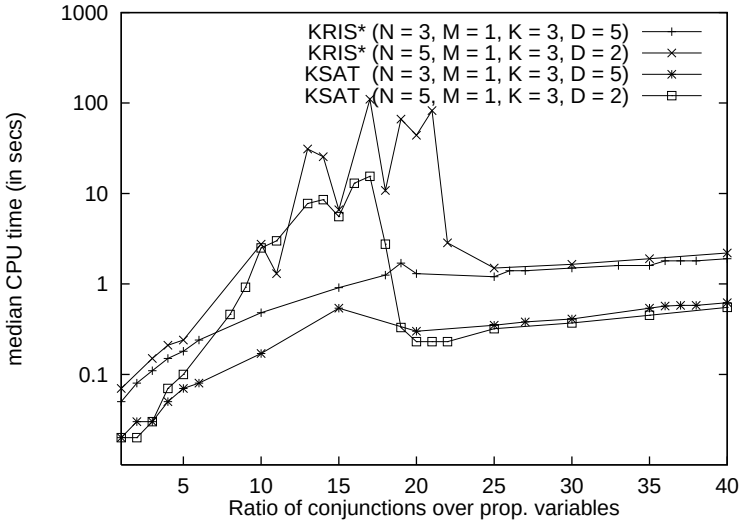


Figure 8: The performance of KSAT and $\mathcal{KRIS}$* for **PS0** and **PS1**

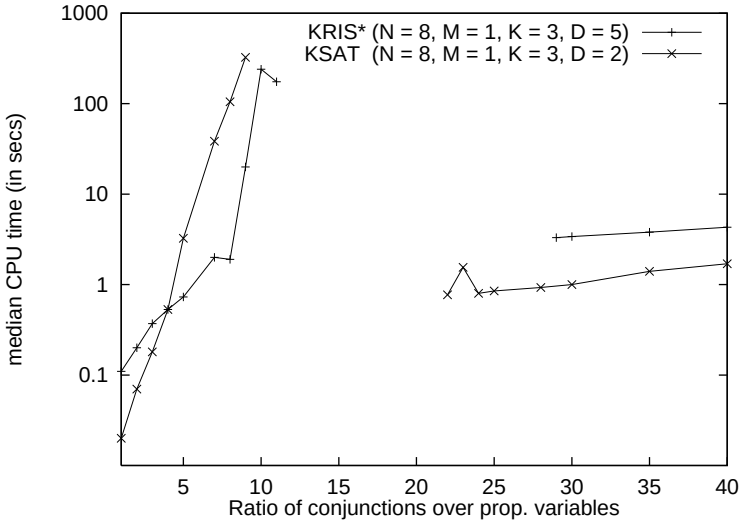


Figure 9: The performance of KSAT and $\mathcal{KRIS}$* for N=8

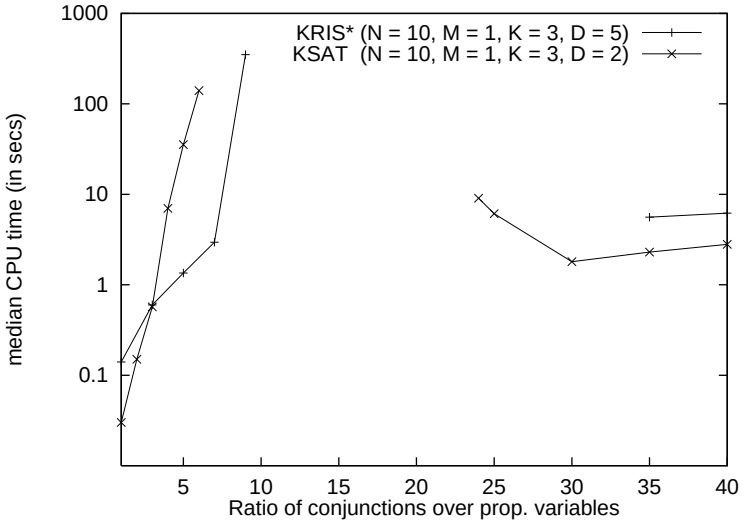


Figure 10: The performance of KSAT and $\mathcal{KRIS}$* for N=10

increased number of intermediate calls to the procedure KM and each call is more expensive than for simpler formulae. The effect is strengthened by the following inefficiency of the intermediate calls to KM. Suppose we have just checked the $K(m)$-satisfiability of the truth assignment $\mu_1 = \{\Box_1\psi_1 = \bot, \Box_1\phi_{11} = \top, \ldots, \Box_1\phi_{1n} = \top\}$ and extend μ_1 by $\{\Box_1\psi_2 = \bot\}$. By the next call to KM, KSAT will not only test whether $\neg\psi_2 \wedge \phi_{11} \wedge \ldots \wedge \phi_{1n}$ is satisfiable, but it will repeat the test whether $\neg\psi_1 \wedge \phi_{11} \wedge \ldots \wedge \phi_{1n}$ is satisfiable. So, KSAT performs the same tests over and over again without need. This could be avoided by memorizing which tests have been done or not.

We now address claim (2) that, intrinsically, SAT-based modal decision procedures are bound to be more efficient than tableaux-based decision procedures. Giunchiglia and Sebastiani base their claim on a result by D'Agostino [D'A 92], who shows that in the worst case, algorithms using the $\vee$`-elim` rule cannot simulate truth tables in polynomial time. Instead one has to use the following modified form of $\vee$`-elim`:

$$\vee\texttt{-elim'}: \quad \frac{w{:}\phi \vee \psi, C \mid S}{w{:}\phi, C \mid w{:}\psi, w{:}\overline{\phi}, C \mid S}$$

This rule ensures that the two subproblems $w{:}\phi, C$ and $w{:}\psi, w{:}\overline{\phi}, C$ generated by the elimination of the disjunction $\phi \vee \psi$ are mutually exclusive.

We have just seen that a major cause of the difference in computational behaviour of the two algorithms is the absence of the preprocessing step in $\mathcal{KRIS}$. To explain the remaining difference we study the quality of the random modal 3CNF formulae. Suppose that we want to test a random modal

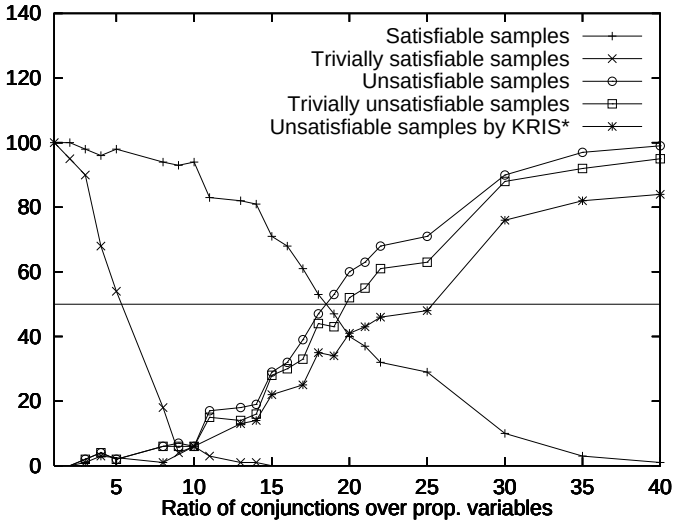


Figure 11: The quality of the test set for **PS0**

3CNF formula ϕ with N propositional variables for satisfiability in a Kripke model with only one world. We have to test at most 2^N truth assignments to the propositional variables. Since $N \leq 5$ for the modal formulae under consideration, this is a trivial task, even by the truth table method. We say a random modal 3CNF formula ϕ is *trivially satisfiable* if ϕ is satisfiable in a Kripke model with only one world. We also say a random modal 3CNF formula ϕ is *trivially unsatisfiable* if the conjunction of the purely propositional clauses of ϕ is unsatisfiable. Again, testing whether ϕ is trivially unsatisfiable requires only the consideration of 2^N truth assignments.

The graphs of Figure 11 show the percentage of satisfiable, trivially satisfiable, unsatisfiable, trivially unsatisfiable, and unsatisfiable formulae in the samples detected by $\mathcal{KRIS}$* of the set of test formulae generated for **PS0**. We see that almost all unsatisfiable test formulae are trivially unsatisfiable. This holds also for all the other parameter settings used by Giunchiglia and Sebastiani. This indicates, none of the parameter settings is suited to generate challenging unsatisfiable modal formulae.

If we consider Figure 8 and 11 together, for ratios L/N between 19 and 21 and N=5 we observe the graph of $\mathcal{KRIS}$* (in Figure 8) deviates a lot (by a factor of more than 100) from the graph of KSAT. This is the area near the crossover point where the percentage of trivially unsatisfiable formulae rises above 50%, however, the percentage of unsatisfiable formulae detected by $\mathcal{KRIS}$* is still below 50% in this area. $\mathcal{KRIS}$* does not detect all trivially unsatisfiable formulae within the time-limit which explains the deviation in performance from KSAT. The reason for $\mathcal{KRIS}$* not detecting all trivially

unsatisfiable formulae within the time limit, can be illustrated by the following example.

Example 5
Let ϕ_4 be a simplified modal 3CNF formula

$$\begin{aligned} & p \wedge q \wedge (m_{11} \vee m_{12} \vee m_{13}) \\ & \quad \ldots \\ & \quad \wedge (m_{k1} \vee m_{k2} \vee m_{k3}) \wedge (\neg p \vee \neg q) \end{aligned}$$

where the m_{ij}, with $1 \leq i \leq k$, $1 \leq j \leq 3$, are modal literals different from p, q, $\neg p$, and $\neg q$. Evidently, ϕ_4 is trivially unsatisfiable. KSAT does the following: Since p and q are unit clauses in ϕ_4, it applies the rule `dp_unit` twice to ϕ. The rule replaces the occurrences of p and q by $\top$, it replaces the occurrences of $\neg p$ and $\neg q$ by $\bot$, and it simplifies the formula. The resulting formula is $\bot$. At this point only the rule `dp_clash` is applicable and KSAT detects that ϕ_4 is unsatisfiable. In contrast, $\mathcal{KRIS}^*$ proceeds as follows. First it applies the $\wedge$`-elim` rule k+2 times, eliminating all occurrences of the $\wedge$ operator. Then it applies the $\vee$`-elim` rule to all disjunctions, starting with $m_{11} \vee m_{12} \vee m_{13}$ and ending with $m_{k1} \vee m_{k2} \vee m_{k3}$. This generates 3^k subproblems. Each of these subproblems contains the literals p and q and the disjunction $\neg p \vee \neg q$. The simplification rule $\vee$`-simp`$_1$ eliminates the disjunction $\neg p \vee \neg q$ and a final application of the $\wedge$`-clash` rule exhibits the unsatisfiability of each subproblem. Obviously, for k large enough, $\mathcal{KRIS}^*$ will not be able to finish this computation within the time-limit.

In the Logics Workbench branch pruning avoids this kind of computation. Starting from the sequent $\Rightarrow \neg\phi_4$ it first applies the $(r\neg)$-rule followed by applications of the $(l\wedge)$-rule until all outer conjunction operators are eliminated. A sequence of k+1 applications of the $(l\vee)$-rule follows generating 2^{k+1} (potential) branches. On the first and second branch the sequents

$$p, q, m_{11}, \ldots, m_{k1}, \neg p \Rightarrow$$

and

$$p, q, m_{11}, \ldots, m_{k1}, \neg q \Rightarrow$$

are considered which are both provable. As neither proof requires the use of one of the literals m_{11}, ... ,m_{k1}, the Logics Workbench prunes all remaining branches and detects the unsatisfiability of ϕ_4.

Note, it makes no difference whether $\mathcal{KRIS}^*$ eliminates disjunctions by the $\vee$`-elim` rule or the $\vee$`-elim'` rule in this example. The reason for $\mathcal{KRIS}^*$ not finishing within the time-limit is that it does not apply the simplification rules $\vee$`-simp`$_1$ and $\vee$`-simp`$_2$ and the $\wedge$`-clash` rule exhaustively before any application of the $\vee$`-elim` rule. So, there is no intrinsic reason that a tableaux-based system cannot outperform KSAT (which is claim (2)). Although the difference between the rules $\vee$`-elim` and $\vee$`-elim'` is fundamental from a theoretical point

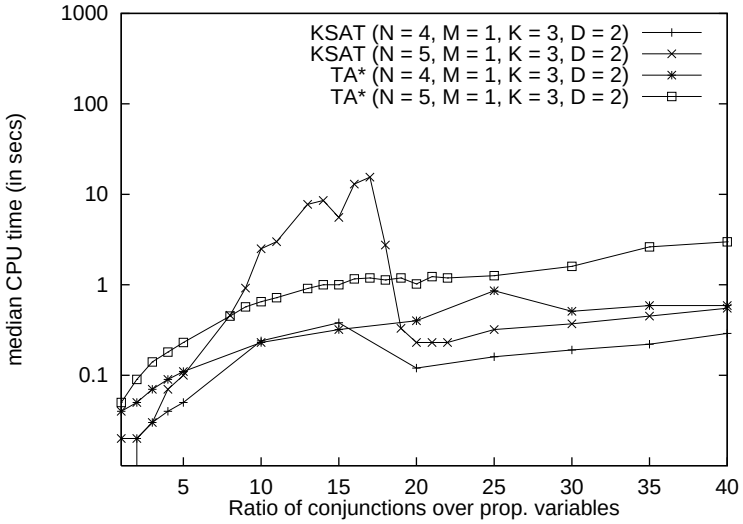


Figure 12: The performance of KSAT and TA*

of view, it is irrelevant on the randomly generated modal formulae under consideration. The reason for $\mathcal{KRIS}$* having worse performance than KSAT is that it has a limited set of simplification rules which are not applied exhaustively before any applications of the branching rule `∨-elim`.

Finally, we consider claim (3) conjecturing an easy-hard-easy pattern, independent of all the parameters of evaluation, in randomly generated modal logic formulae. We have seen in Figure 5 that the median CPU time consumption of KSAT decreases drastically at the ratio $L/N = 17.5$ for the second sample. This is almost the point, where 50% of the sample formulae are satisfiable. This decline seems to resemble the behaviour of propositional SAT decision procedures on randomly generated 3SAT problems. Figure 12 compares the performance of KSAT with the performance of the translation approach on two parameter settings, where the easy-hard-easy pattern is most visible for KSAT. The translation approach does not show the peaking behaviour of KSAT. The median CPU time grows monotonically with the size of modal formulae. Thus, the phase transition visible in Figure 5 is an artificial phenomenon of KSAT (and $\mathcal{KRIS}$), and not an intrinsic property of the generated modal formulae, which refutes conjecture (3).

Observe that the peaking behaviour occurs in the area where the number of trivially satisfiable sample formulae approaches zero. The following example tries to explain this.

Example 6
Let ϕ_5 be a simplified modal 3CNF formula of the form

$$\neg\Box_1 s \wedge \Box_1(p \vee r) \wedge (\Box_1\neg r \vee \Box_1 q) \wedge (\neg\Box_1 p \vee \Box_1 r) \wedge (m_{11} \vee m_{12} \vee m_{13}) \\ \ldots \\ \wedge (m_{n1} \vee m_{n2} \vee m_{n3})$$

where the m_{ij}, with $1 \leq i \leq n$, $1 \leq j \leq 3$, are modal literals different from the modal literals in the first four conjunctions of ϕ_5. Let us assume that ϕ_5 is satisfiable. Observe:

1. $\Box_1\neg r$ is false in any model of ϕ_5, since $\Box_1\neg r$ and $\neg\Box_1 s \wedge (\neg\Box_1 p \vee \Box_1 r)$ imply $\neg\Box_1 p$, and $\Box_1(p \vee r) \wedge \Box_1\neg r \wedge \neg\Box_1 p$ is not $K(m)$-satisfiable.
2. As a consequence, any truth assignment μ such that $\mu(\Box_1\neg r) = \top$ is not $K(m)$-satisfiable.
3. A unit propagation step by KDP, replacing $\Box_1\neg r$ by $\top$, does not affect the literal $\Box_1 r$.

KSAT starts by assigning $\top$ to $\neg\Box_1 s$ and $\Box_1(p \vee r)$. Then it will apply a sequence of applications of the `dp_split` and `dp_unit` rules to ϕ_5. Let us assume that the first split variable is $\Box_1\neg r$, followed by k modal literals $m_1, \ldots, m_k$ chosen from $m_{11}, \ldots, m_{n3}$, and finally $\neg\Box_1 p$. Before any further applications of the `dp_split` rule, KSAT calls the procedure KM to test the $K(m)$-satisfiability of the current truth assignment μ. Since μ assigns $\top$ to $\Box_1\neg r$, KM will fail. However, KSAT has no means to detect the primary cause of the failure. KSAT continues by considering all other cases generated by the application of `dp_split` to $\neg\Box_1 p$, m_k, m_{k-1}, $\ldots$, m_1. It will fail to generate a satisfying truth assignment in all these cases. Finally, it considers the case that $\Box_1\neg r$ is false. Eventually, KSAT finds a satisfying truth assignment to ϕ_5. However, KSAT has considered at least 2^{k+1} cases unnecessarily without finding a satisfying truth assignment. This explains the bad behaviour of KSAT on those sample formulae where satisfiability tests in the non-propositional context are essential. $\mathcal{KRIS}$* behaves even worse since it delays the application of the $\Diamond_i$`-elim` until no other rule can be applied.

By contrast, the Logics Workbench takes advantage of its branch pruning. Starting from the sequent $\Rightarrow \neg\phi_5$ it first applies the $(r\neg)$-rule followed by applications of the $(l\wedge)$-rule until all outer conjunction operators are eliminated. A sequence of applications of the $(l\vee)$-rule follows. Let us assume the disjunctions are considered in this order: $(\Box_1\neg r \vee \Box_1 q)$, $(m_{11} \vee m_{12} \vee m_{13})$, $\ldots$, $(m_{n1} \vee m_{n2} \vee m_{n3})$, and finally $(\neg\Box_1 p \vee \Box_1 r)$. Like for KSAT and $\mathcal{KRIS}$ this generates 2^{n+2} (potential) branches. On the first branch the sequent Γ_1

$$\Box_1(p \vee r), \Box_1\neg r, m_{11}, \ldots, m_{n1} \Rightarrow \Box_1 p, \Box_1 s$$

is considered. This sequent is provable. So, the Logics Workbench considers the second branch generated by the application of the $(l\vee)$-rule to $(\neg\Box_1 p \vee \Box_1 r)$.

Again, the sequent Γ_2

$$\Box_1(p \vee r), \Box_1 \neg r, m_{11}, \ldots, m_{n1}, \Box_1 r \Rightarrow \Box_1 s$$

is provable. As neither the proof of Γ_1 nor of Γ_2 makes use of any of the literals $m_{11}, \ldots, m_{n1}$, the Logics Workbench does branch pruning. It will jump back directly to the point where the ($l\vee$)-rule is applied to $(\Box_1 \neg r \vee \Box_1 q)$ and considers the branch in which $\Box_1 q$ is added to the left-hand side of the sequent. Thus, the search space is reduced considerably.

The translation approach proceeds as follows. It generates a clause set for ϕ_5 containing the five clauses

$$\begin{array}{l} \mathit{def}_1 \\ \neg S(\underline{a}) \\ \neg \mathit{def}_1 \vee P_1(x) \vee R_1(x), \\ \neg \mathit{def}_1 \vee \neg R_1(x) \vee \neg \mathit{def}_1 \vee Q_1(y), \\ \neg P_1(\underline{b}) \vee \neg \mathit{def}_1 \vee R_1(x) \end{array}$$

where $\underline{a}$ and $\underline{b}$ denote Skolem constants associated with the two occurrences of $\neg\Box_1$ and x and y are variables. Unit propagation of the first clause followed by subsumption replaces the original clause set by the following one:

$$\begin{array}{l} \mathit{def}_1 \\ \neg S_1(\underline{a}) \\ P_1(x) \vee R_1(x), \\ \neg R_1(x) \vee Q_1(y), \\ \neg P_1(\underline{b}) \vee R_1(x) \end{array}$$

Three resolvents can be derived from these clauses: $P_1(x) \vee Q_1(y)$, $\neg P_1(\underline{b}) \vee Q_1(y)$, and $R_1(\underline{b}) \vee R_1(x)$. Factoring on the last resolvent yields the unit clause $R(\underline{b})$. At this point, the translation approach has detected that $\Box_1 \neg r$ is not satisfiable in any model of ϕ_5. An additional inference step computes the unit clause $Q_1(y)$. No further inference is possible on this subset.

Using the splitting inference rule of SPASS it is also possible to construct a derivation which resembles closely the one of the Logics Workbench. Instead of computing the three resolvents we can start by splitting the clause $\neg R_1(x) \vee Q_1(y)$ into its variable-disjoint subclauses, $\neg R_1(x)$ and $Q_1(y)$. Let us first consider the branch on which we add the clause $\neg R_1(x)$ to the clause set. This corresponds to assigning true to $\Box_1 \neg r$. Let us assume that the translation of the disjunctions $(m_{11} \vee m_{12} \vee m_{13})$ to $(m_{n1} \vee m_{n2} \vee m_{n3})$ (indicated by a * below) generates clauses to which we can apply the splitting rule as well. Finally, apply the splitting rule to $\neg P_1(\underline{b}) \vee R_1(x)$. On the first branch we

consider the clause set $\mathcal{S}_1$

$$
\begin{array}{l}
\quad def_1 \\
\neg S_1(\underline{a}) \\
\quad P_1(x) \vee R_1(x), \\
\neg R_1(x), \\
\quad m^*_{11}, \\
\quad \cdots, \\
\quad m^*_{n1}, \\
\neg P_1(\underline{b}).
\end{array}
$$

The clauses $\neg P_1(\underline{b})$, $P_1(x) \vee R_1(x)$, and $\neg R_1(x)$ yield a contradiction. Since the clause introduced by the last application of the splitting rule is involved in the derivation of the empty clause, we have to consider the clause set $\mathcal{S}_2$

$$
\begin{array}{l}
\quad def_1 \\
\neg S_1(\underline{a}) \\
\quad P_1(x) \vee R_1(x), \\
\neg R_1(x), \\
\quad m^*_{11}, \\
\quad \cdots, \\
\quad m^*_{n1}, \\
\quad R_1(x).
\end{array}
$$

Here, $\neg R_1(x)$ and $R_1(x)$ produce a contradiction. Since none of the clauses $m^*_{11}, \ldots, m^*_{n1}$ have been used in the refutation of $\mathcal{S}_1$ and $\mathcal{S}_2$, branch condensing will prevent the consideration of any of the alternative branches that exist for these clauses. SPASS proceeds directly by considering the branch where the clause $Q_1(y)$ belongs to the set of clauses.

Examples 5 and 6 illustrated how the branch pruning technique of the Logics Workbench can avoid two pitfalls in which $\mathcal{KRIS}$* and KSAT can be caught. Figure 13 shows however that brunch pruning alone does not lead to an improved median CPU time consumption for all formulae. The following example illustrates what happens.

Example 7
Consider the formula ϕ_6

$$
\begin{array}{l}
\quad p \vee \Box_1 q \\
\wedge \neg \Box_1 (p \vee r \vee q) \vee p \\
\wedge \Box_1 (p \vee q) \vee \Box_1 (q \vee p) \\
\wedge \Box_1 p \vee \Box_1 q \vee p.
\end{array}
$$

$\mathcal{KRIS}$* will easily detect the satisfiability of ϕ_6. After exhaustive application of the conjunction elimination rule, it applies $\vee$`-elim` to the first disjunction $p \vee \Box_1 q$, $\vee$`-simp`$_0$ to the second disjunction, $\vee$`-elim` to the third disjunction

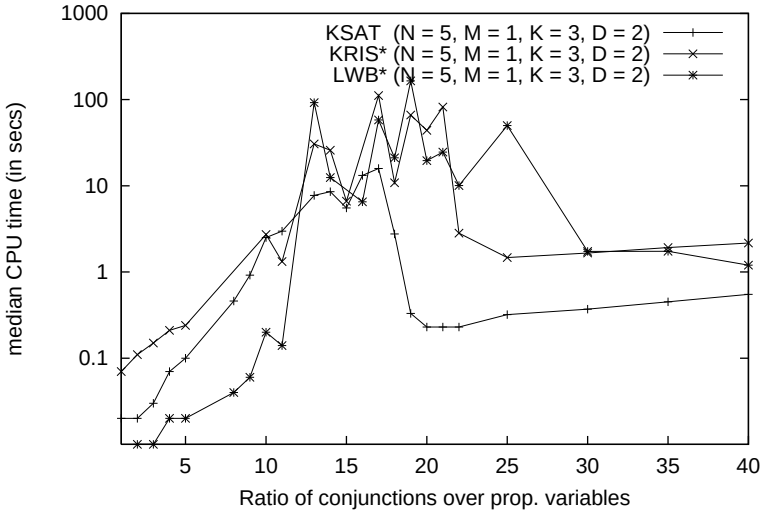


Figure 13: The performance of the Logics Workbench

and $\vee$-simp$_0$ to the fourth disjunction. $\mathcal{KRIS}$* obtains the set of labelled formulae $\{w_0{:}p, w_0{:}\Box_1(p \vee q)\}$ to which no further rules can be applied.

The Logics Workbench has no equivalent to $\vee$-simp$_0$ and deals with the second and fourth disjunction by means of the $(l\vee)$-rule. Furthermore, it will consider the left branch introduced by an (backwards) application of the $(l\vee)$-rule, first. So, it considers the sequent Γ_1

$$p, \neg\Box_1(p \vee r \vee q), \Box_1(p \vee q) \vee \Box_1(q \vee p), \Box_1 p \vee \Box_1 q \vee p \Rightarrow$$

before Γ_2

$$p, p, \Box_1(p \vee q) \vee \Box_1(q \vee p), \Box_1 p \vee \Box_1 q \vee p \Rightarrow$$

which are both obtained from $\Rightarrow \neg\phi_6$, by applications of the $(r\neg)$-, $(l\wedge)-$, and $(l\vee)$-rules. After further applications of the $(l\vee)$- and $(r\Box_i)$-rule, the Logics Workbench discovers that Γ_1 is provable and turns to Γ_2. Only then it detects that ϕ_6 is satisfiable.

So, the Logics Workbench spends a serious amount of computational effort considering obviously useless branches introduced by the $(l\vee)$-rule. Figure 13 seems to indicate that this overwhelms the gain of branch pruning. It is worth noting that the behaviour of the Logics Workbench on KCNF formulae can be improved either by adding simplification rules or by employing better criteria for selecting the branches introduced by the $(l\vee)$-rule.

Example 7 also illustrates that is important to first assign a truth value to the propositional variables in a random formula since this allows to reduce the number of further assignments.

7 Broadening the evaluation

The graphs of the previous sections and of the papers of Giunchiglia and Sebastiani are 50% percentile graphs as each point presents the median CPU time consumption for 100 formulae with ratio L/N. This means that the graphs merely reflect the performance for the easier half of the formulae set. More informative are the collections of 50%, 60%, ... , 100% percentile graphs we present in Figures 14(a), 14(b), 14(c) and 14(d). Formally, the Q%-*percentile* of a set of data is the value V such that Q% of the data is smaller or equal to V and $(100-Q)$% of the data is greater than V. The median of a set coincided with its 50%-percentile.

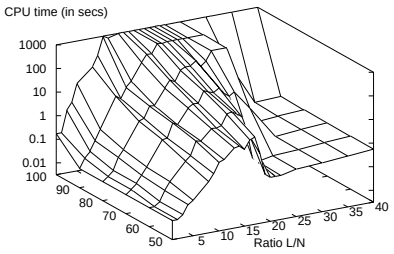
The Figures 14(a), 14(b), 14(c) and 14(d) respectively show the percentile graphs for KSAT, $\mathcal{KRIS}$*, LWB* and the translation approach on the parameter setting **PS0** (N=5, M=1, K=3, D=2). The difference in shape for KSAT, $\mathcal{KRIS}$*, and the Logics Workbench as opposed to that for the translation approach is striking.

For the translation approach the difference between the 50%-percentile and the 90%-percentile of the CPU time consumption is marginal. We see the same monotonic increase of the CPU time consumption with increasing ratio L/N for all percentiles. Only the 100%-percentile reaches the time-limit of 1000 CPU seconds at some points. This means, there are some hard random 3CNF formulae in the collection, but for each ratio L/N their number does not exceed 10. This again supports our view that the problems generated using the parameter settings **PS0** are easier than the computational behaviour of KSAT and the other methods except the translation approach indicates.

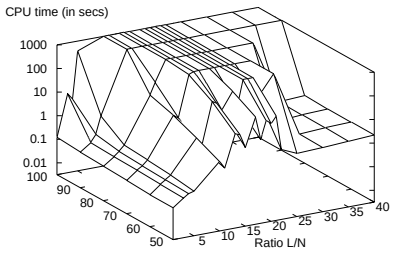
The contrast to $\mathcal{KRIS}$ and the Logics Workbench is most extreme. While the Logics Workbench shows a good uniform behaviour where the ratio L/N is smaller than 10, we see a dramatic breakdown for ratios L/N greater than 10. As the percentage of trivially satisfiable samples reaches zero, the Logics Workbench can hardly complete 60% of the sample formulae within the time-limit. Even at ratios L/N above 30 where the percentage of trivially unsatisfiable formulae is greater than 90%, the Logics Workbench fails on 10% of the formulae. Similarly, for $\mathcal{KRIS}$. The absence of simplification rules in the Logics Workbench explains the less prominent 'valley' for ratios L/N above 30.

The percentage of sample formulae on which a decision procedure fails to complete its computation within a given time-limit (of reasonable size) may be regarded as a kind of risk for the user of that decision procedure. We call this the *failure risk*. The failure risk for each procedure is reflected in Figures 14(a) to 14(d) by the size of the plateau at the time-limit of 1000 CPU seconds. The risk of failure for the parameter setting under examination is highest for the Logics Workbench and $\mathcal{KRIS}$*, and lowest for the translation approach.

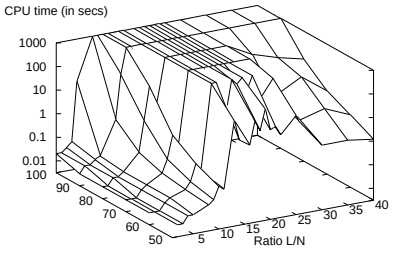
We call the percentage of sample formulae on which a decision procedure terminates its computation within a given time-limit the *success chance* of a decision procedure. The notions of success chance and failure risk are complementary. The success chance will be regarded as an additional measure of the



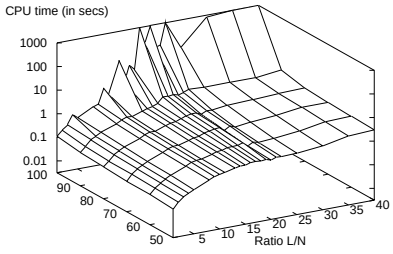
(a) KSAT



(b) $\mathcal{KRIS}$*



(c) The Logics Workbench



(d) The translation approach

Figure 14: Percentile graphs for **PS0**

quality of a decision procedure. The weighting of the two quality measures, the success chance and median CPU time consumption, depends on the preferences of the user.

The percentile graphs are more informative and provide a better framework for comparison than the median curves. We can say KSAT performs better than $\mathcal{KRIS}^*$ and has a higher chance of success on the entire range of ratios L/N for the parameter setting **PS0**. The Logics Workbench is unbeatable for ratios L/N below 7. We believe the graphs indicate a qualitative difference in the performance of the translation approach as opposed to the other three theorem provers.

8 Where the hard problems are

This section considers the question of how the parameter settings and random formula generator can be modified to provide better (more difficult) test samples.

The parameter setting **PS0** provides the most challenging collection of random 3CNF formulae among all the parameter settings used by Giunchiglia and Sebastiani. The Figures 15 and 16 show the influence of the parameter N, that is, the number of propositional variables, on the median CPU time consumption of KSAT and the translation approach. We see an increasing median CPU time consumption over the range of the ratio L/N with increasing value N. Thus increasing the number of propositional variables involved in the random generation of modal 3CNF formula provides more challenging test samples.

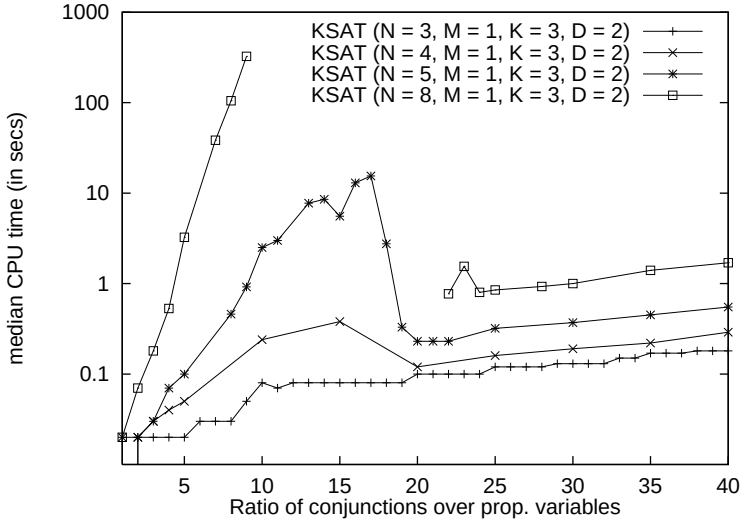


Figure 15: Varying the parameter N for KSAT

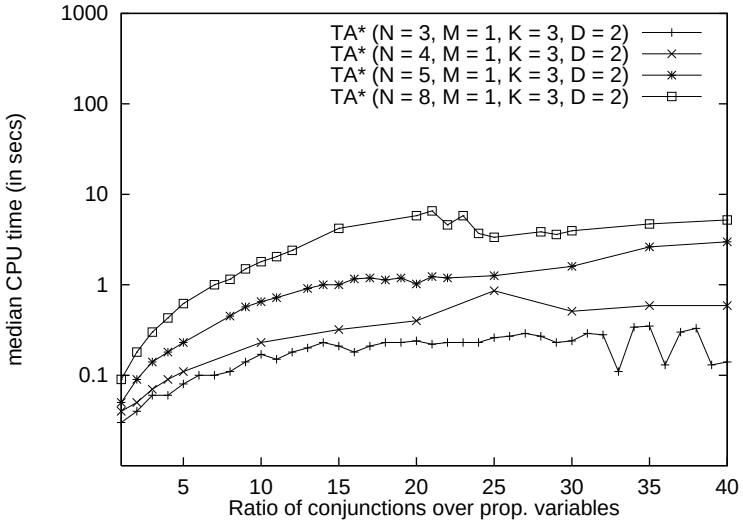


Figure 16: Varying the parameter N for the translation approach

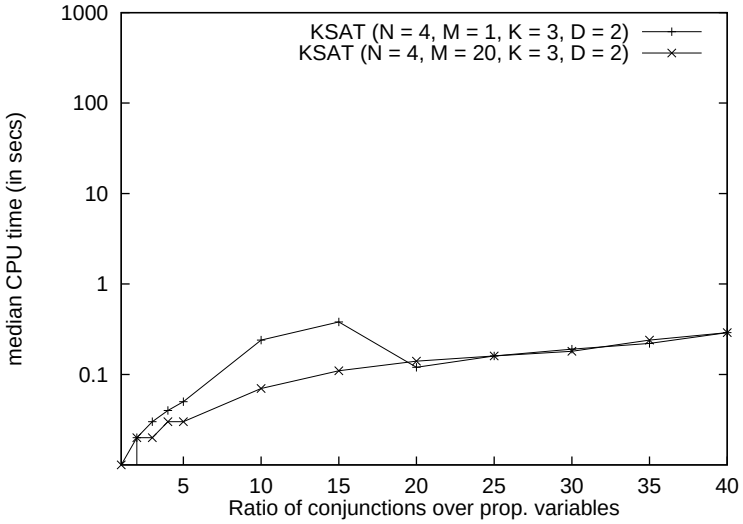


Figure 17: Varying the parameter M for KSAT

The Figures 17 and 18 provide an indication of the influence of the parameter M, that is, the number of modalities, on the median CPU time consumption of KSAT and the translation approach. The influence on the translation approach can be considered as being insignificant. Likewise we see that for a ratio

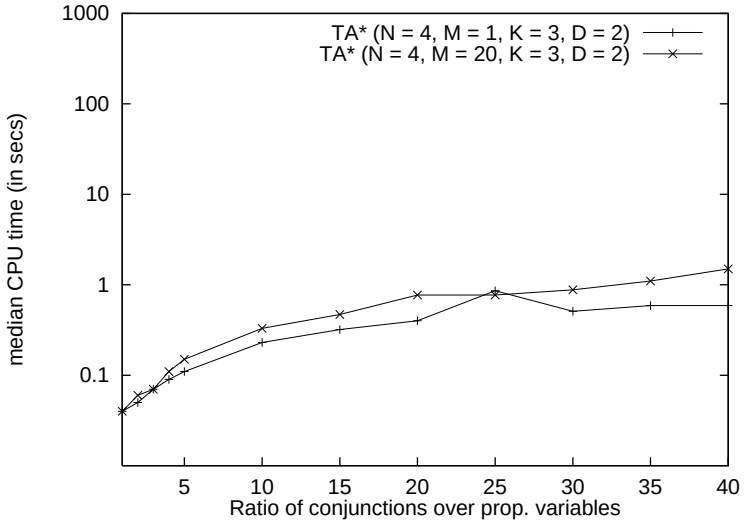


Figure 18: Varying the parameter M for the translation approach

L/N greater than 20, the median CPU time consumption of KSAT on the two parameter settings are identical. This can be explained by our observation that almost all unsatisfiable formulae are trivially unsatisfiable. The modal subformulae in trivially unsatisfiable formulae are irrelevant. Therefore, increasing the number of modalities is also irrelevant for unsatisfiable formulae. Below a ratio L/N of 20, the modal formulae generated using only one modality seem to be slightly more challenging than the modal formulae generated using twenty different modalities. This is due to the fact that the procedure KM is less likely to fail for twenty modalities than for just one modality [GS 96b]. The small divergence in the behaviour of KSAT on **PS5** (N=4, M=1, D=2, P=0.5) and **PS9** (N=4, M=20, D=2, P=0.5) is due to a smaller number of contradictions between modal literals for **PS9**. We illustrate this observation by the following example.

Example 8
The formula ϕ_7 given by

$$(\Box_1(p \vee q) \vee \Box_1(r \vee q)) \wedge \neg\Box_1(q \vee p \vee s)$$

is satisfiable. KSAT will first apply the `dp_unit` rule replacing $\Box_1(q \vee p \vee s)$ by $\bot$. The first conjunct of ϕ_7 is left unchanged and KSAT has to apply the `dp_split` rule. Suppose it chooses $\Box_1(p \vee q)$ as split 'variable'. Replacing $\Box_1(p \vee q)$ by $\top$ renders ϕ_7 true propositionally, but checking the satisfiability of $\neg(q \vee p \vee s) \wedge (p \vee q)$ reveals that this truth assignment is not $K(m)$-satisfiable. So we have to continue with $\Box_1(r \vee q)$, the second case generated by the `dp_split` rule.

Replacing the last remaining modal atom by $\top$ again renders the formula true propositionally. Finally, we have to check the satisfiability of $\neg(q \vee p \vee s) \wedge (r \vee q)$ which succeeds.

In contrast consider the formula ϕ_8 given by

$$(\Box_2(p \vee q) \vee \Box_1(r \vee q)) \wedge \neg\Box_1(q \vee p \vee s),$$

which is like ϕ_7 except the first occurrence of a $\Box_1$ is replaced by $\Box_2$. KSAT proceeds in the same way as for ϕ_7. It replaces $\Box_1(q \vee p \vee s)$ by $\bot$ and chooses $\Box_2(p \vee q)$ as split 'variable'. Replacing $\Box_2(p \vee q)$ by $\top$ renders ϕ true propositionally. But now instead of checking the satisfiability of $\neg(q \vee p \vee s) \wedge (p \vee q)$ we just have to check that $\neg(q \vee p \vee s)$ is satisfiable, because $p \vee q$ occurs below a different modality. Since this check succeeds ϕ_8 is satisfiable. Evidently, the computation for ϕ_8 is easier than for ϕ_7.

Now we vary the parameter D, the modal depth of the randomly generated modal 3CNF formulae. The situation for the parameter D is slightly more complicated than for the parameters N and M. By the definition of modal 3CNF formulae, increasing the modal depth increases the size of the formulae. The size, however, is an important factor influencing the performances of the procedures under consideration. Although the graphs in Figures 20 and 19 seem to indicate that increasing the modal depth of the sample formulae also increases the median CPU time consumption of the decision procedures, the increase parallels the increase of the median size of the modal formulae shown in Figure 21. A closer look at the graphs reveals that increasing the modal depth of the randomly generated modal 3CNF formulae actually makes the satisfiability problem easier. While the median formula size increases by a factor of five between modal depth 2 and modal depth 5, the median CPU time consumption of KSAT only increases by a factor of three.

Based on these observations we identify three guidelines for generating more challenging problems.

1. Parameters that have no significant influence on the "difficulty" of the randomly generated formulae should be set to the smallest possible value. This applies to the parameters M and D. That is, we restrict our attention to random modal 3CNF formulae of degree one using only one modality.

2. We have to avoid generating trivially unsatisfiable modal formulae. A straightforward solution is to require that all literals of a 3CNF clause of modal degree 1 are expressions of the form $\Box_1\phi$ or $\neg\Box_1\phi$ where ϕ is a random modal 3CNF clause of propositional variables. This amounts to setting the parameter P to zero.

3. For all occurrences of $\Box_1\phi$ in a random modal 3CNF formula of degree 1, ϕ has to be a non-tautologous clause containing exactly three differing literals.

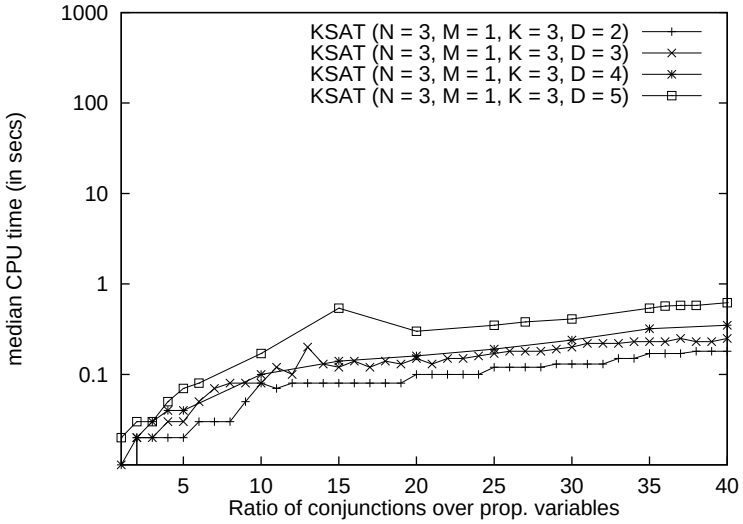


Figure 19: Varying the parameter D for KSAT

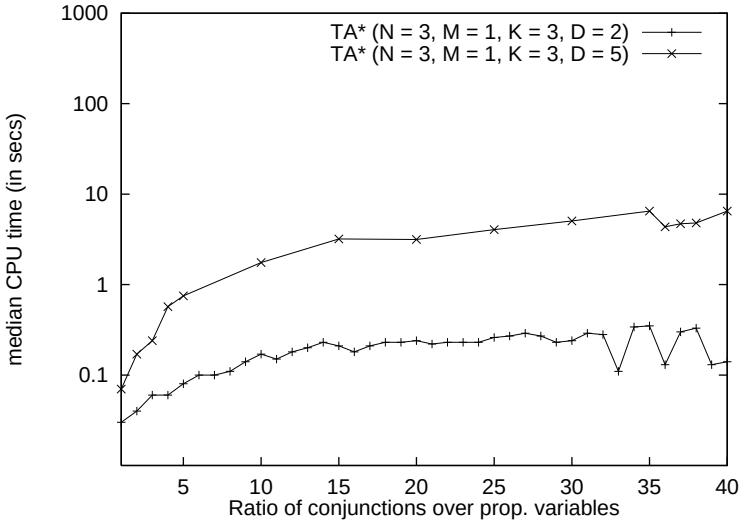


Figure 20: Varying the parameter D for the translation approach

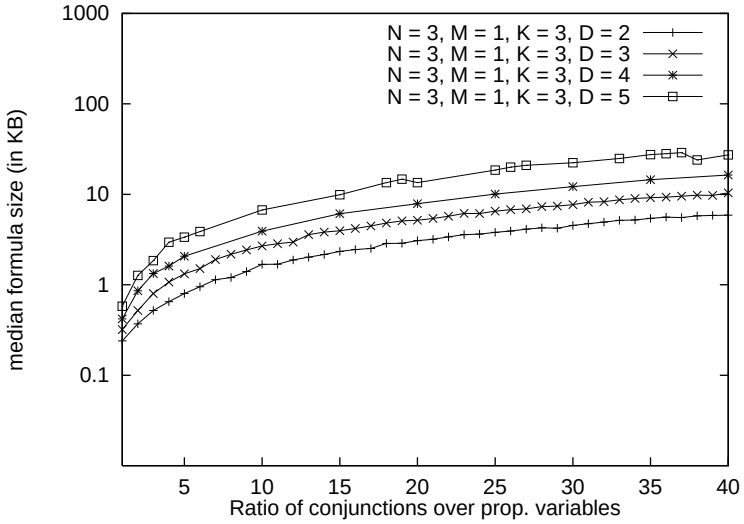


Figure 21: The influence of the parameter D on the formula size

In line with the second guideline one may consider excluding also trivially satisfiable modal formulae. However, this amounts to doing preliminary satisfiability checks of the generated modal formulae in order to identify and reject the trivially satisfiable ones. For the moment, we do not perform these checks.

The restriction to random modal 3CNF formulae of degree one is somewhat surprising if one takes into account that if we bound D then the worst-case complexity of the satisfiability problem in basic modal logic is no longer PSPACE-complete, but NP-complete. This is a point that deserves further investigation. How can difficult modal formulae with increased modal degree be generated automatically? Some difficult examples of higher degree which have been constructed by hand can be found in the benchmark collection of the Logics Workbench [HS 96].

The parameters not fixed by the three guidelines are the number N of propositional variables and the number K of literals in any clause. We choose to fix K=3 in two parameter settings **PS12** (N=4, M=1, K=3, D=1, P=0) and **PS13** (N=6, M=1, K=3, D=1, P=0). Figure 22 reflects the quality of the parameter setting **PS12** by the percentage of satisfiable, unsatisfiable, trivially satisfiable, and trivially unsatisfiable modal formulae in the sample sets we generated. Compared to Figure 11 (page 23) for the parameter setting **PS0**, the percentage of trivially satisfiable formulae has decreased significantly. As expected, the percentage of trivially unsatisfiable formulae is zero. Furthermore, the percentage of satisfiable formulae decreases faster. Already for the ratio L/N of 25 there are almost no satisfiable formulae. For this reason, the experiments consider only the sample sets with ratio L/N between 1 and 30.

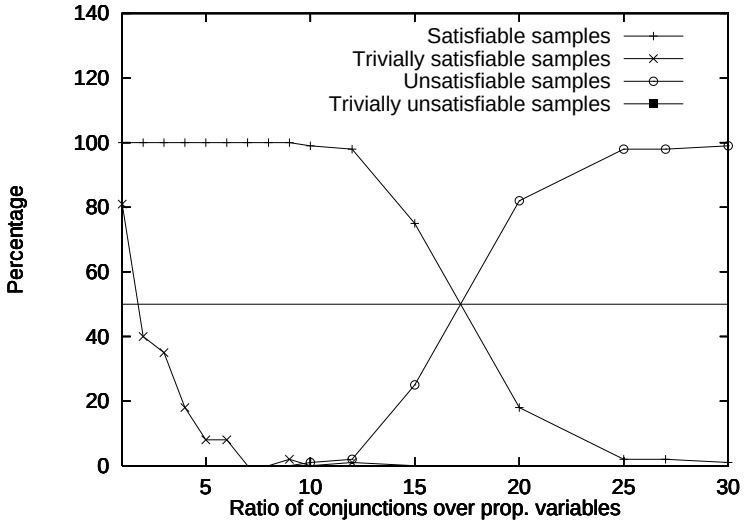


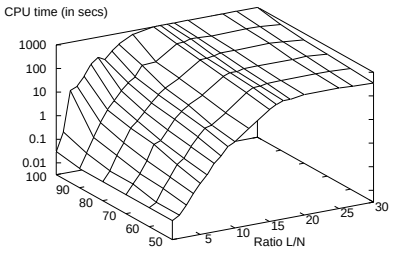
Figure 22: The quality of the test set for **PS12**

The percentile graphs of KSAT, $\mathcal{KRIS}$*, LWB* and the translation approach on the settings **PS12** are given in Figures 23(a) to 23(d). Again, we observe that KSAT outperforms $\mathcal{KRIS}$* and the Logics Workbench, while the translation approach does best. More important, the formulae generated by the new parameter settings and the modified random generator are much harder than any of the formula samples generated for the settings **PS0** to **PS9** by the original generator. Figures 24(a) to 24(d) show the percentile graphs on **PS13**. We see that even the translation approach fails to decide within the given time-limit the satisfiability of half of the input formulae for ratios L/N greater than 13. The failure rates of KSAT, $\mathcal{KRIS}$*, and the Logics Workbench are even higher.

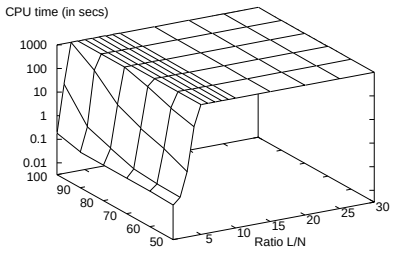
Conclusion

We have pointed out a number of problems with evaluating the performance of different algorithms for modal reasoning. A crucial factor is the quality of the randomly generated formulae. Even for propositional theorem proving defining adequate random formula generators for performance evaluation is hard [CI 95]. We have shown that the random generator and parameter settings used in [GS 96a, GS 96b] produce formulae with particular characteristics (like redundant subformulae, almost no non-trivially unsatisfiable formulae within the test sets) which have to be carefully taken into account in an empirical study. We give some guidelines for modifying the generator.

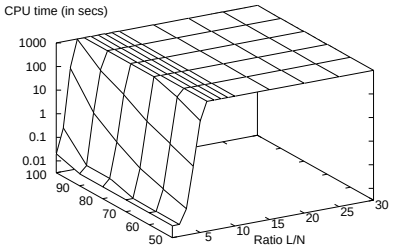
The basic algorithm of KSAT is very similar to the algorithms of $\mathcal{KRIS}$ and



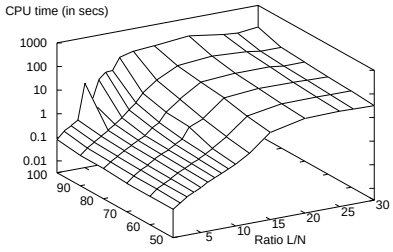
(a) KSAT



(b) $\mathcal{KRIS}$*

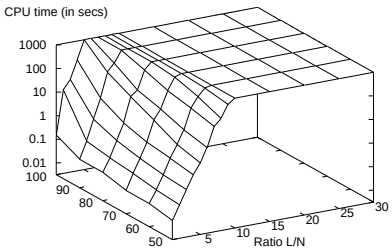


(c) The Logics Workbench

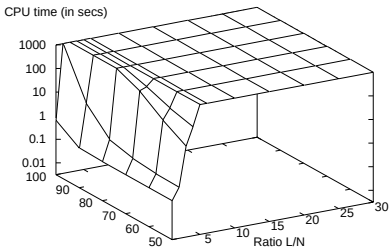


(d) The translation approach

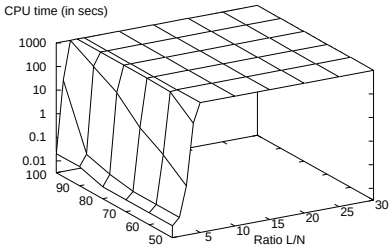
Figure 23: Percentile graphs for **PS12**



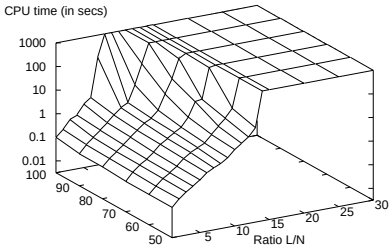
(a) KSAT



(b) $\mathcal{KRIS}$*



(c) The Logics Workbench



(d) The translation approach

Figure 24: Percentile graphs for **PS13**

the Logics Workbench. The essential differences between KSAT, $\mathcal{KRIS}$, and the Logics Workbench are:

1. KSAT utilizes an elaborated set of simplification rules for boolean and modal formulae. These are the `dp_unit` inference rule of the procedure KDP and the rules in Table 1. These rules are applied whenever possible throughout the computation. By contrast, $\mathcal{KRIS}$ has only a very limited set of simplification rules, namely $\vee$`-simp`$_0$ and $\vee$`-simp`$_1$, which are applied occasionally. The Logics Workbench uses no simplification rules at all.

2. KSAT utilizes a heuristic for selecting the particular disjunction for the application of disjunction elimination (namely, applying `dp_split` to a modal atom with a maximal number of occurrences). By contrast, $\mathcal{KRIS}$ and the Logics Workbench process disjunctions in a fixed order determined by the ordering of the disjunctions in the input formula.

3. KSAT performs intermediate checks of $K(m)$-satisfiability of the current truth assignment before every application of the `dp_split` rule. This corresponds to an application of our proposed $\Diamond_i$`-test` inference rule for tableaux-based systems. $\mathcal{KRIS}$ has no equivalent of the $\Diamond_i$`-test` rule.

   The Logics Workbench has a similar strategy as $\mathcal{KRIS}$. It delays the application of the $(l\Diamond_i)$- and the $(r\Box_i)$-rules until no further applications of the other rules are possible.

4. The Logics Workbench utilizes branch pruning which the other systems do not.

Based on our performance evaluation and the insights we have gained by inspecting the code of the various systems under examination, our assessment of the relevance of these differences between the theorem provers concerning their performance is the following:

1. The presence of simplification rules and their exhaustive application is vital for any theorem prover, particularly for the class of formulae we have been considering. It is surprising that there are theorem provers like $\mathcal{KRIS}$ and the Logics Workbench making very little use of simplification.

2. Further investigations will have to answer whether elaborated heuristics for the selection of split 'variables' in the application of the `dp_split` rule or disjunctions in the application of the $\vee$-elimination rule lead to improved performance of KSAT for the entire range of generated sample sets.

3. The introduction of intermediate calls to the KM procedure to check the $K(m)$-satisfiability of the current truth assignment is valuable. It makes a difference to the performance of KSAT. However, in its present form

KSAT cannot make optimal use of the information provided by a failure of an intermediate call to KM (Example 6).

We envisage that more redundancy can be eliminated by delaying the application of rules dealing with modal operators and using branch pruning to backtrack to an appropriate state of the search space, like the Logics Workbench does.

Further improvements of the SAT-based procedure KSAT are possible and further investigations are needed to evaluate the usefulness of the various techniques. All the techniques can be transferred to tableaux-based systems like $\mathcal{KRIS}$ and sequent calculus-based systems like the Logics Workbench. Likewise the techniques employed in $\mathcal{KRIS}$ and the Logics Workbench can be transferred to KSAT.

Our experiments show that the translation approach in combination with the theorem prover SPASS has better computational behaviour than KSAT, $\mathcal{KRIS}$, and the Logics Workbench on all samples of randomly generated modal 3CNF formulae we have considered, except for the samples of very small or easily solvable formulae. This is due to the initial overhead of the transformation to clausal form. It is open which resolution inference rules and search strategies perform best for basic modal logic and its extensions. We emphasize the positive results of this paper obtained for the combination of the translation approach and SPASS can most probably not be obtained with less sophisticated theorem provers (without splitting and branch condensing). The question now is whether, with the proposed optimisation and simplification techniques, the special purpose theorem provers can achieve similar performance as the translation approach, or possibly do better.

Acknowledgements

We thank the developers of KSAT, $\mathcal{KRIS}$, and the Logics Workbench for making available the code of their systems. The critical comments of Christoph Weidenbach, Andreas Nonnengart and the anonymous referees helped improve the paper considerably. This work was conducted while the authors were employed at the Max-Planck-Institut für Informatik in Saarbrücken, Germany.

References

[BdlT 92] T. Boy de la Tour. An optimality result for clause form translation. *Journal of Symbolic Computation*, 14:283–301, 1992.

[BG 90] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In M. E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction (CADE-10)*, LNAI 499, pages 427–441, Springer, 1990.

[BH 91] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In H. Boley and M. M. Richter, editors, *Proceedings of the International Workshop on Processing Declarative Knowledge (PDK '91)*, LNAI 567, pages 67–86, Springer, 1991.

[Cat 91] L. Catach. TABLEAUX: A general theorem prover for modal logics. *Journal of Automated Reasoning*, 7(4):489–510, 1991.

[CI 95] B. Cha and K. Iwama. Performance test of local search algorithms using new types of random CNF formulas. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 304–311, Morgan Kaufmann, 1995.

[D'A 92] M. D'Agostino. Are tableaux an improvement on truth-tables? *Journal of Logic, Language, and Information*, 1:235–252, 1992.

[Fit 83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Company, 1983.

[Gob 74] L. F. Goble. Gentzen systems for modal logic. *Notre Dame Journal of Formal Logic*, 15:455–461, 1974.

[GS 96a] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures: Case study of modal K. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 583–597, Springer, 1996.

[GS 96b] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 304–314, Morgan Kaufmann, 1996.

[HJSS 96] A. Heuerding, G. Jäger, S. Schwendimann, and M. Seyfried. The Logics Workbench LWB: A snapshot. *Euromath Bulletin*, 2(1):177–186, 1996.

[HS 96] 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.

[HS 97] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logics. In M. E. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 202–207, Morgan Kaufmann, 1997.

[MSL 92] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465, MIT Press, 1992.

[Ohl 91] H. J. Ohlbach. Semantics based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[Ohl 93] H. J. Ohlbach. Translation methods for non-classical logics: An overview. *Bulletin of the IGPL (Interest Group in Propositional and Predicate Logics)*, 1(1):69–90, 1993.

[OS 97] H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. *Journal of Logic and Computation*, 7(5):581–603, October 1997.

[Pet 83] G. E. Peterson. A technique for establishing completeness results in theorem proving with equaility. *SIAM Journal of Computation*, 12(1):82–100, February 1983.

[Rau 83] W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12:403–423, 1983.

[Sch 98a] R. A. Schmidt. Decidability by resolution for propositional modal logics. To appear in *Journal of Automated Reasoning*, 1998.

[Sch 98b] R. A. Schmidt. Resolution is a decision procedure for many propositional modal logics. In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic, Volume 1*, volume 87 of *Lecture Notes*, pages 189–208. CSLI Publications, 1998.

[SSS 91] M. Schmidt-Schauß and G. Smolka. Attributive concept description with complements. *Artifical Intelligence*, 48:1–26, 1991.

[Wei 96] C. Weidenbach. *Computational Aspects of a First-Order Logic with Sorts*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[Wei 97] C. Weidenbach. SPASS Version 0.49. *Journal of Automated Reasoning*, 18(2):247–252, 1997. For a more recent version see `http://www.mpi-sb.mpg.de/guide/software/spass.html`.