MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

On evaluating decision procedures for modal logic

Ullrich Hustadt and Renate A. Schmidt

MPI–I–97–2–003 April 1997

INFORMATIK

Im Stadtwald
D 66123 Saarbrücken
Germany

Authors' Addresses

Ullrich Hustadt
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
`hustadt@mpi-sb.mpg.de`
`http://www.mpi-sb.mpg.de/~hustadt`

Renate A. Schmidt
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
`schmidt@mpi-sb.mpg.de`
`http://www.mpi-sb.mpg.de/~schmidt`

Acknowledgements

We thank Christoph Weidenbach and Andreas Nonnengart for their critical comments. The work of the first author is partially supported by the DFG grant Ga 261/8-1. The work of the second author is supported by the TraLos-Project funded by the DFG.

Abstract

This paper reports on 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 [8, 9]. 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 has superior computational behaviour compared to the other three approaches.

Keywords

Modal theorem provers, empirical performance analysis, random modal formulae

1 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 $\mathcal{ALC}$ [19]. Some approaches utilize standard first-order theorem proving techniques in combination with translations from propositional modal logic to first-order logic [14, 15]. Others use Gentzen systems [10, 11]. Still others use tableaux proof methods [7, 17, 1].

Usually, the literature on theorem provers for modal logic confines itself to a description of the underlying calculus and methodology accompanied with a consideration of the worst-case complexity of the resulting algorithm. Sometimes a small collection of benchmarks is given as in [4]. There have not been any exhaustive empirical evaluations or comparisons of the computational behaviour of theorem provers based on different methodologies.

Giunchiglia and Sebastiani [8, 9] changed that. They report on an exhaustive empirical analysis of the tableaux system $\mathcal{KRIS}$ [1] 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 [11], a system based on a Gentzen-style calculus for modal logic and the functional translation approach of Ohlbach [13]. 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 mislead Giunchiglia and Sebastiani in their analysis and comparison of KSAT and the tableaux system $\mathcal{KRIS}$. The random generator does not produce challenging unsatisfiable modal formulae. So as to obtain harder problems we develop guidelines for 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 2, 3, 4 and 5 briefly describe the inference mechanisms of KSAT, $\mathcal{KRIS}$, the Logics Workbench and the translation approach. Section 6 defines random modal formulae and describes the test method of Giunchiglia and Sebastiani, which Section 7 evaluates. Section 8 presents percentile graphs for the four systems that are more informative than graphs presenting the median CPU time consumption. Finally, Section 9 proposes improvements to the random generator so as to produce more challenging random samples on which the methods are tested.

2 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 $\neg\Box_i\neg\psi$.

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 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 that 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 each φ_{ij} is satisfiable, the formula ϕ is $K(m)$-satisfiable. If at least one of the formulae φ_{ij} is not satisfiable, then KM is said to *fail on* μ.

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.

$$\texttt{dp\_sol:} \quad \frac{\langle \top,\, \mu \rangle \mid P \triangleright S}{P \triangleright S \cup \{\mu\}}$$

$$\texttt{dp\_clash:} \quad \frac{\langle \bot,\, \mu \rangle \mid P \triangleright S}{P \triangleright S}$$

$$\texttt{dp\_unit:} \quad \frac{\langle \phi[c],\, \mu \rangle \mid P \triangleright S}{\langle \phi',\, \mu \cup \{c = \top\} \rangle \mid P \triangleright S}$$

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.

$$\texttt{dp\_split:} \quad \frac{\langle \phi[m],\, \mu \rangle \mid P \triangleright S}{\langle \phi[m] \wedge p,\, \mu \rangle \mid \langle \phi[m] \wedge \neg p,\, \mu \rangle \mid P \triangleright S}$$

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

The symbol $\mid$ denotes concatenation of sequences. $\overline{\phi}$ denotes the complementary formula of ϕ, for example $\overline{\neg p} = p$ and $\overline{\Box_i p} = \Diamond_i \neg p$.

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 [8, pp. 583-584] suggest that KSAT0 can be based on any decision procedure for propositional logic. However, this is false.

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

$$\begin{aligned}\phi_1 = {} & (p \vee q \vee \neg\Box_1(p \vee \neg p))\wedge \\ & (\neg p \vee \neg q \vee \neg\Box_1(p \vee \neg p)).\end{aligned}$$

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.

By way of one example, we will illustrate the four satisfiability testing approaches under consideration.

Example 1:
Consider the satisfiable modal formula $\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. Figure 1 depicts the derivation tree of KSAT for this formula. 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

Figure 1: Sample derivation of KSAT

$$\begin{tikzcd}
 & \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q) \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_unit}"'] & & \\
 & \Box_1 p \vee \Box_1 q \arrow[dl, Rightarrow] \arrow[dr, Rightarrow, "\textsf{KDP: }\texttt{dp\_split}"'] & & \\
\Box_1 p \wedge (\Box_1 p \vee \Box_1 q) \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_unit}"'] & & \neg\Box_1 p \wedge (\Box_1 p \vee \Box_1 q) \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_unit}"] & \\
\top \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_sol}"'] & & \Box_1 q \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_unit}"] & \\
\begin{array}{c} \mu_1 = \{\Box_1(p \vee r) = \bot, \\ \Box_1 p = \top\} \end{array} \arrow[d, Rightarrow, "\textsf{KM}"'] & & \top \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_sol}"] & \\
\neg(p \vee r) \wedge p \arrow[d, Rightarrow, "\textsf{KDP: }\texttt{dp\_unit}"'] & & \begin{array}{l} \mu_2 = \{\Box_1(p \vee r) = \bot, \\ \quad \Box_1 p = \bot,\ \Box_1 q = \top\} \end{array} \arrow[d, Rightarrow] \arrow[dr, Rightarrow, "\textsf{KM}"'] & \\
\bot & & \neg(p \vee r) \wedge q \arrow[d, Rightarrow] & \neg p \wedge q \arrow[d, Rightarrow] \\
 & & \top & \top
\end{tikzcd}$$

simplified to $\Box_1 p \vee \Box_1 q$ 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.

3 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 labeled 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 $\{w{:}\psi\} \cup C$.

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

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

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

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

$\vee$-`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.

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 simplification rules are applied only immediately before an application of the $\vee$-`elim` rule and then they are applied only to the labeled 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 labeled 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 labeled 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 preprocessing procedure that Giunchiglia and Sebastiani use in conjunction with KSAT which we discuss later (see Table 1 on page 21).

The condition that the $\Diamond_1$-`elim` rule can be applied only if none of the other rules can be applied to the set of labeled 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 the sequence

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

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 labeled 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 labeled 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 labeled 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 labeled 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.

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

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 modal 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 labeled 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 labeled 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 labeled formula reveals that our set of labeled formulae contains both $w_1{:}\neg p$ and $w_1{:}p$. This is a contradiction and the $\wedge$`-clash` rule eliminates this set of labeled formula from the sequence. The remaining set of labeled 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 $w_1{:}\neg p \wedge \neg r \wedge p$ 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 labeled formulae to which no further rule applies. Thus, $\mathcal{KRIS}$ has shown that ψ' and ψ are satisfiable.

4 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 [11] (of which some axioms and rules are eliminable).

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

$$\begin{tikzcd}[row sep=large]
w_0{:}\Diamond_1(\neg p \wedge \neg r) \wedge (\Box_1 p \vee \Box_1 q) \arrow[d, Rightarrow, "\wedge\texttt{-elim}"] \\
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p \vee \Box_1 q \arrow[d, Rightarrow, "\vee\texttt{-elim}"] \\
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p \mid w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \arrow[d, Rightarrow, "\Diamond\texttt{-elim}"] \\
w_1{:}(\neg p \wedge \neg r) \wedge p, w_0{:}\Box_1 p \mid w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \arrow[d, Rightarrow, "\wedge\texttt{-elim}"] \\
w_1{:}\neg p, w_1{:}\neg r, w_1{:}p, w_0{:}\Box_1 p \mid w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \arrow[d, Rightarrow, "\wedge\texttt{-clash}"] \\
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \arrow[d, Rightarrow, "\Diamond\texttt{-elim}"] \\
w_1{:}(\neg p \wedge \neg r) \wedge q, w_0{:}\Box_1 q \arrow[d, Rightarrow, "\wedge\texttt{-elim}"] \\
w_1{:}\neg p, w_1{:}\neg r, w_1{:}q, w_0{:}\Box_1 q
\end{tikzcd}$$

Axioms:

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

Rules:

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

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

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

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

Figure 3: Axioms and rules of the Logics Workbench

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.

The Logics Workbench does not support multiple modalities, although the rules $(l\Diamond)$ and $(r\Box)$ can be generalized easily.

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:

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

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 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 $\vee\texttt{-simp}_1$ 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 $(l\Diamond)$-rule and $(r\Box)$-rule are applied only after no application of the other rules is possible.

Example 3:
Figure 4 gives the proof 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 $(r\Box)$-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

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

Figure 4: Sample derivation of the Logics Workbench

sequent, the choice is deterministic. The application of the $(r\Box)$-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 $(r\Box)$-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, branch pruning in the Logics Workbench, and the conversion to negation normal form by $\mathcal{KRIS}$.

In Section 7 we restrict ourselves to a comparison of the Ksat, $\mathcal{KRIS}$, and the translation approach. However, due to the similarity between the Logics Workbench and $\mathcal{KRIS}$, our comments concerning $\mathcal{KRIS}$ also apply to the Logics Workbench.

5 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 [15]. It has the property that ordinary resolution without any refinement strategies is a decision procedure for $K(m)$ [18]. 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 $\neg de_i$ 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 $\forall\beta\exists\alpha\,\psi$'. Ohlbach and Schmidt prove $\Upsilon\Pi_f$ preserves satisfiability, more specifically, ϕ is a theorem in $K(m)$ if and only if $\neg\Upsilon\Pi_f(\phi)$ is unsatisfiable.

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 basic multi-modal logic 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. 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 in 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.

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

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 developed by Weidenbach *et al.* [21]. 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 [3] 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 [2] extended with the sort techniques of Weidenbach [20].

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 [16].

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.

4. It supports dynamic sort theories by additional inference rules including sort generation and sort resolution and additional reduction rules like sort simplification and clause deletion.

Ordered inference rules and splitting are of particular importance when treating satisfiable formulae. 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.

6 The evaluation method

The evaluation method adopted by Giunchiglia and Sebastiani follows the approach of Mitchell *et al.* [12]. To set up a benchmark suite for Davis-Putnam-based theorem provers Mitchell *et al.* [12] 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 form 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. They fix all parameters except L, the number of clauses in a formula. For example, they choose N=3, M=1, K=3, D=5, and P=0.5. 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 they measure the time needed by one of the decision procedures to determine the satisfiability of ϕ. Since checking a single formula can take arbitrarily long in the worst case, there is an upper limit for the CPU time consumed. 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). Our tests have been run on a Sun Ultra 1/170E with 196MB main memory using a time-limit of 1000 CPU seconds. The gaps in the graphs (for example for $\mathcal{KRIS}$ above $L/N = 5$) indicate that more than 50 out of 100 formulae of given ratio L/N had to be abandoned.

Giunchiglia and Sebastiani [9] present graphs for the following parameter

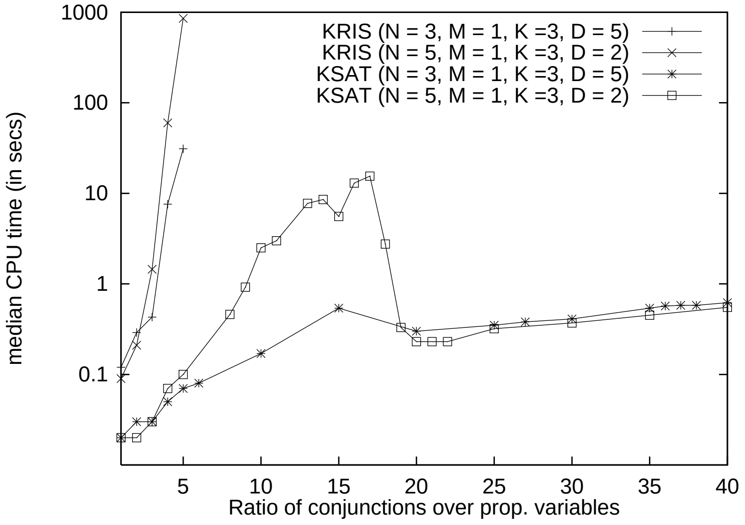

Figure 5: The performance of $\mathcal{KRIS}$ and KSAT for **PS0** and **PS1**

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 [9, 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 9.

7 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 evaluate the quality of the test instances first, before starting a performance comparison. This is particularly important when we set up a completely new collection of test instances. We address the question of whether the random generator and the parameter settings chosen by Giunchiglia and Sebastiani [8, 9] are appropriate for this purpose and actually support 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 (as mistakenly claimed by Giunchiglia and Sebastiani [9, p. 307]). 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 negate 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 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 con-

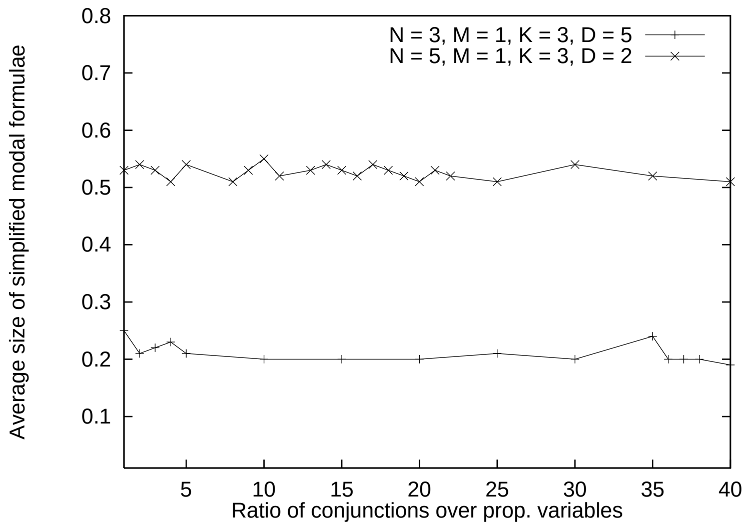

Figure 6: The effect of simplifying modal 3CNF formulae

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

Table 1: The simplification rules of KSAT

tradictory 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 syntactically equal to $\Box_1(q \vee p)$. KSAT also sorts conjunctions and disjunctions alphabetically, 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 disabled the reordering of clauses.

KSAT performs exactly the simplification whose effect is illustrated in Figure 6. $\mathcal{KRIS}$, on the other hand, does not. We consider how KSAT performs if we remove the preprocessing step from its code. In Figure 7 KSAT0 denotes this modified form of KSAT (KSAT0 coincides with the algorithm described

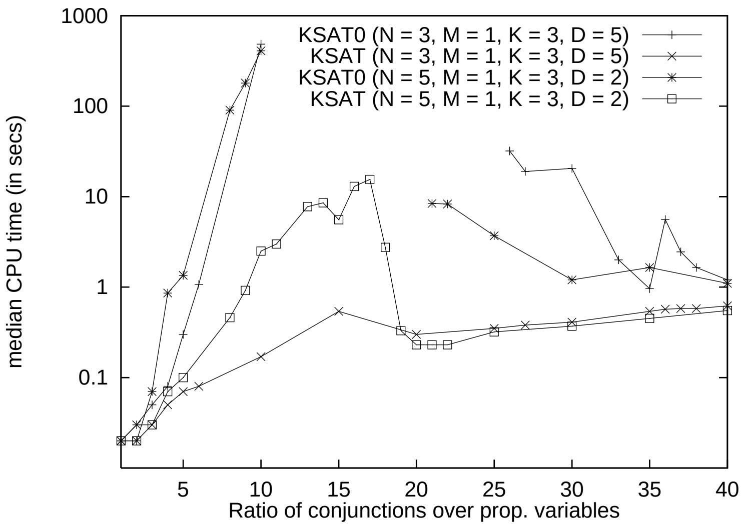

Figure 7: The performance of Ksat and Ksat0 for **PS0** and **PS1**

in Section 2). We see that the behaviour of Ksat0 differs from the behaviour of Ksat by orders of magnitude. In particular, for ratios L/N between 11 and 20 on **PS0**, and between 11 and 25 on **PS1** Ksat0 is no longer able to determine the satisfiability of half of the input formulae.

Since the preprocessing is not an intrinsic part of the decision procedures, for the comparison of the procedures, either both Ksat and $\mathcal{KRIS}$ should utilize the preprocessing or none of them should. Simplification of the generated modal formulae is reasonable, so we have added the preprocessing function to $\mathcal{KRIS}$. This modified version of $\mathcal{KRIS}$ will be denoted by $\mathcal{KRIS}$*. The graphs in Figure 8 show the performances of Ksat and $\mathcal{KRIS}$*. Although the performance of Ksat is still better than that of $\mathcal{KRIS}$*, Ksat is no longer qualitatively better than $\mathcal{KRIS}$ with preprocessing.

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

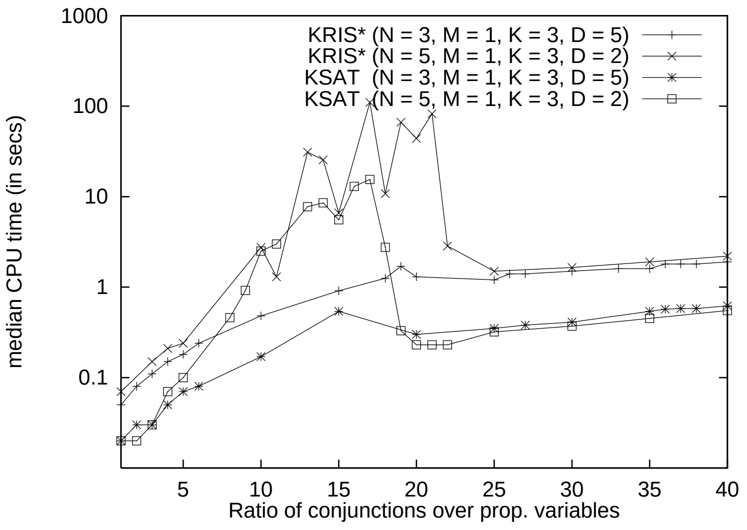

Figure 8: The performance of KSAT and $\mathcal{KRIS}^*$ for **PS0** and **PS1**

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 increased number of intermediate calls to the procedure KM and

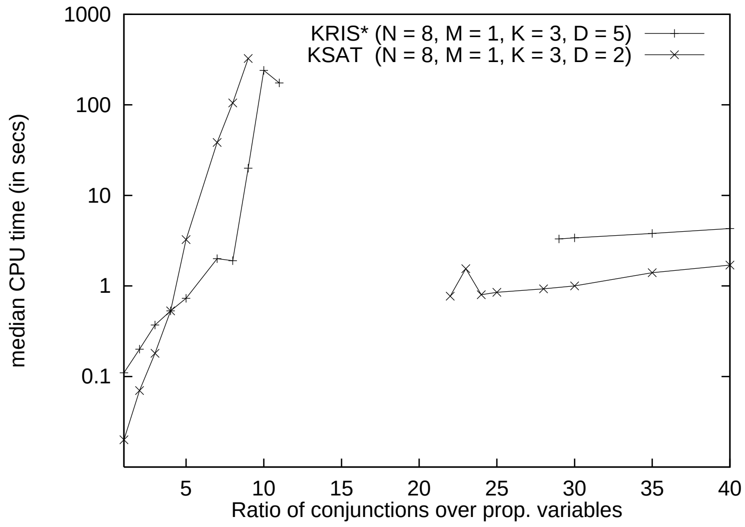

Figure 9: The performance of KSAT and $\mathcal{KRIS}^*$ for N=8

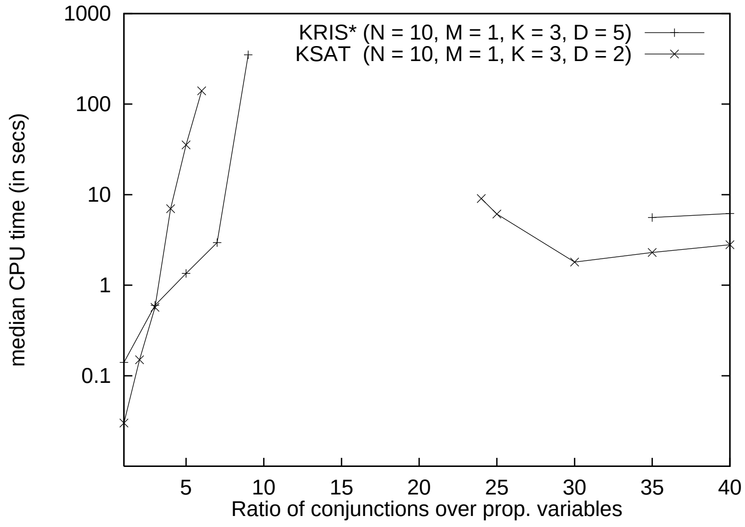

Figure 10: The performance of KSAT and $\mathcal{KRIS}^*$ for N=10

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.

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 the work by D'Agostino [6], who shows that in the worst case, algorithms using the `∨-elim` rule cannot simulate truth tables in polynomial time. Instead one has to use the following modified form of `∨-elim`:

$$\text{∨-elim'}: \quad \frac{w{:}\phi \vee \psi, C \mid S}{w{:}\phi, C \mid w{:}\psi, w{:}\neg\phi, C \mid S}$$

This rule ensures that the two subproblems $w{:}\phi, C$ and $w{:}\psi, w{:}\neg\phi, C$ 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 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

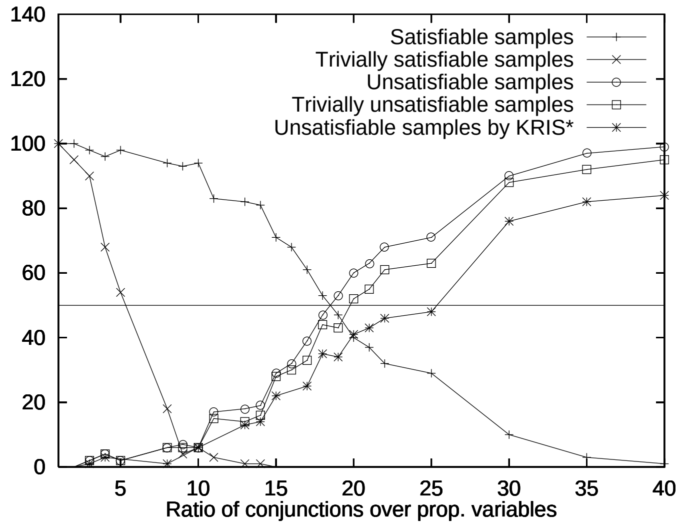

Figure 11: The quality of the test set for **PS0**

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

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

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, 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 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.)

Note, it makes no difference whether $\mathcal{KRIS}^*$ eliminates disjunctions by the $\vee$`-elim` rule or the $\vee$`-elim'` rule. 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.

Similarly, we are able to explain the behaviour of $\mathcal{KRIS}$ as compared with KSAT0 depicted in Figures 5 and 7. KSAT0 has a much better performance on those sample formulae which are more likely unsatisfiable. Suppose according to our parameter setting we have five different propositional variables occurring in the random modal 3CNF formulae. KSAT0, like KSAT, uses the propositional or modal atom occurring most often as the splitting 'variable' for the application of the `dp_split` rule. This has the effect that KSAT will first use all the propositional variables in the formula as splitting variables before using any of the modal atoms. After applying unit propagation we end up with at most 2^N formulae which consist of modal literals of degree greater than 0 only. Any further step of KDP assigns a truth value to one of the modal literals. Before any application of the `dp_split` rule, KM will check the $K(m)$-satisfiability of the current truth assignment. As soon as we have assigned $\top$ to a modal literal m which can be reduced to $\bot$ using the simplification rules in Table 1, this check will fail. KDP will not continue to extend the current assignment, but will go on with a different formula generated by `dp_split`. Based on the graph in Figure 6 we can assume that there is a near 50% chance that a modal atom is $K(m)$-unsatisfiable. So, the intermediate checks by KM will often fail thereby cutting down the number of cases generated by the `dp_split` rule.

In contrast, $\mathcal{KRIS}$ will first use the $\vee$-elimination rule to generate about 3^L sets of labeled formulae where L is the number of clauses in the modal 3CNF formula under consideration. Then it proceeds with $\Diamond_i$-elimination which will reveal the unsatisfiability of each of the sets of labeled formulae. Since L may well exceed 100, there is no hope that $\mathcal{KRIS}$ finishes within the time-limit.

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 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 $\vee$`-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. As in the case of $\mathcal{KRIS}^*$ the preprocessing routine of KSAT has been added to the translation approach which is indicated in the figures by 'TA*'. 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.

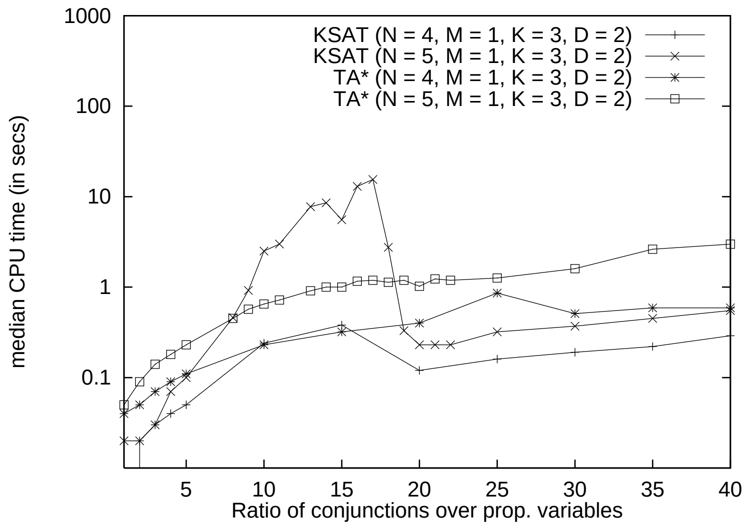

Figure 12: The performance of KSAT and TA*

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

$$\begin{aligned} \neg\Box_1 s \wedge \Box_1(p \vee r) &\wedge (\Box_1\neg r \vee \Box_1 q) \\ &\wedge (\neg\Box_1 p \vee \Box_1 r) \\ &\wedge (m_{11} \vee m_{12} \vee m_{13}) \\ &\dots \\ &\wedge (m_{n1} \vee m_{n2} \vee m_{n3}) \end{aligned}$$

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 three 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 one of the first split variables 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.

In contrast, 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 $R_1(\underline{b}) \vee R_1(x)$. A factoring step on the last resolvent yields the unit clause $R(\underline{b})$. At this point, the translation approach has detected that $\Box_1\neg r$

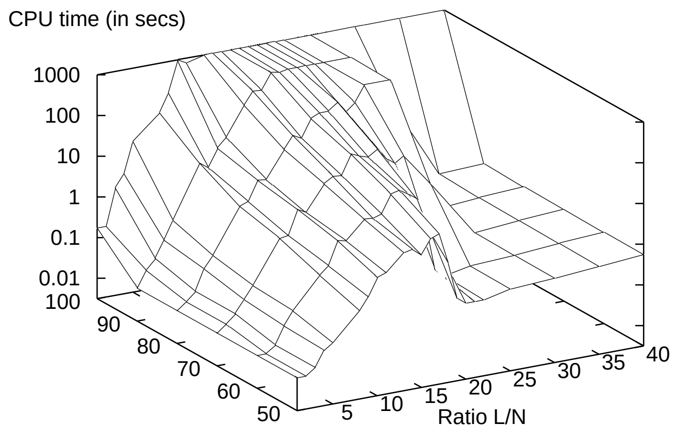

Figure 13: The percentile graphs for KSAT on **PS0**

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.

8 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. More informative are the collections of 50%, 60%, ... , 100% percentile graphs we present in Figures 13, 14, 15 and 16. 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 13, 14, 15 and 16 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). LWB* is the Logics Workbench plus the preprocessing routine that we also included in $\mathcal{KRIS}$*. In the remainder of the section, the Logics Workbench is assumed to include preprocessing. The difference in shape for KSAT, $\mathcal{KRIS}$*, and the Logics Workbench as opposed to that for the translation approach is striking.

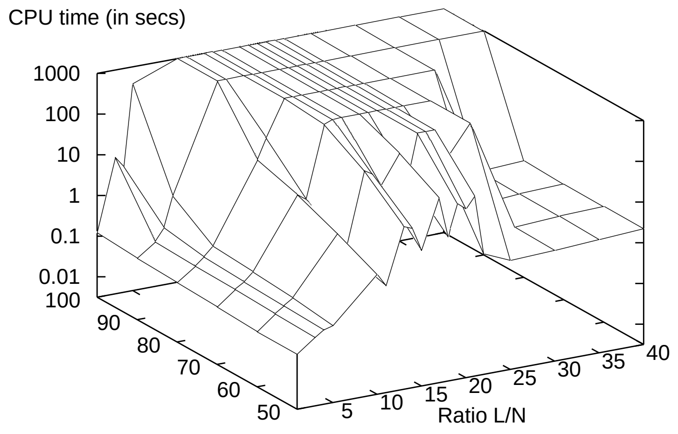

Figure 14: The percentile graphs for $\mathcal{KRIS}^*$ on **PS0**

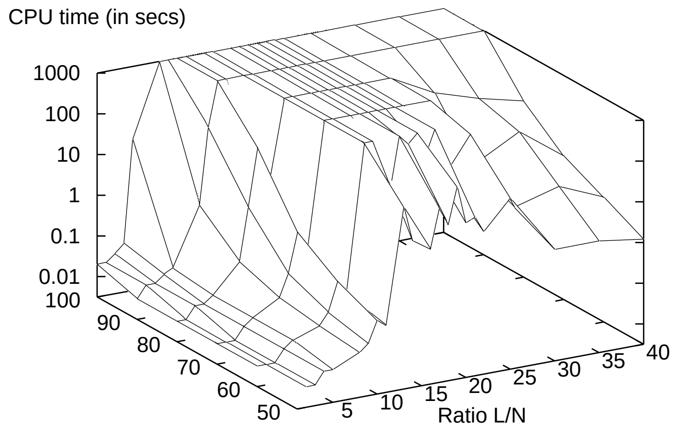

Figure 15: The percentile graphs for LWB* on **PS0**

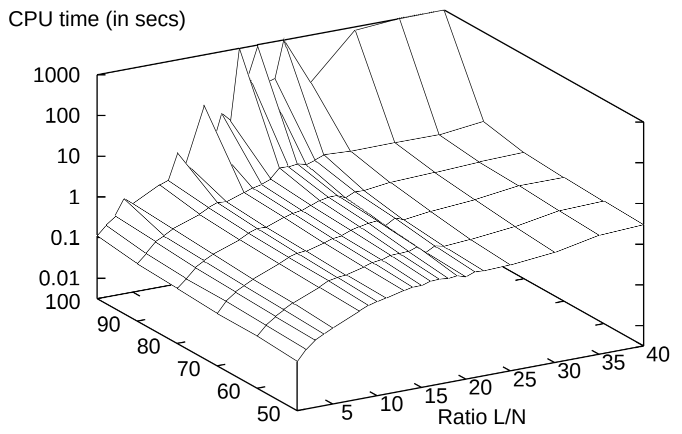

Figure 16: The percentile graphs for the translation approach on **PS0**

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 parameter settings of Giunchiglia and Sebastiani are not adequate to produce challenging problems.

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 13 to 16 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 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 approaches.

9 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 17 and 18 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.

The Figures 19 and 20 provide an indication of the influence of the parameter M, that is, the number of modalities, on the median CPU time

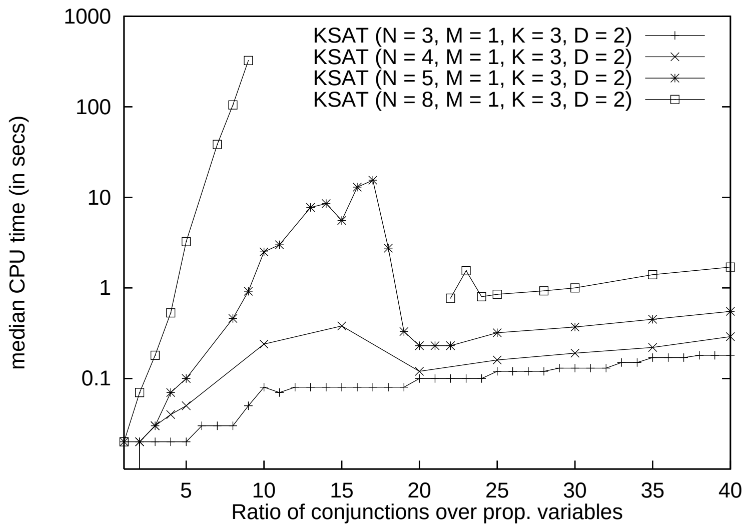

Figure 17: Varying the parameter N for KSAT

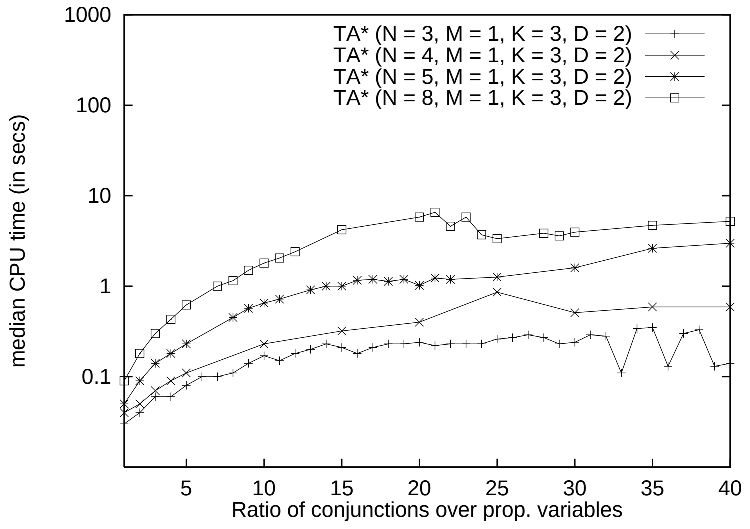

Figure 18: Varying the parameter N for the translation approach

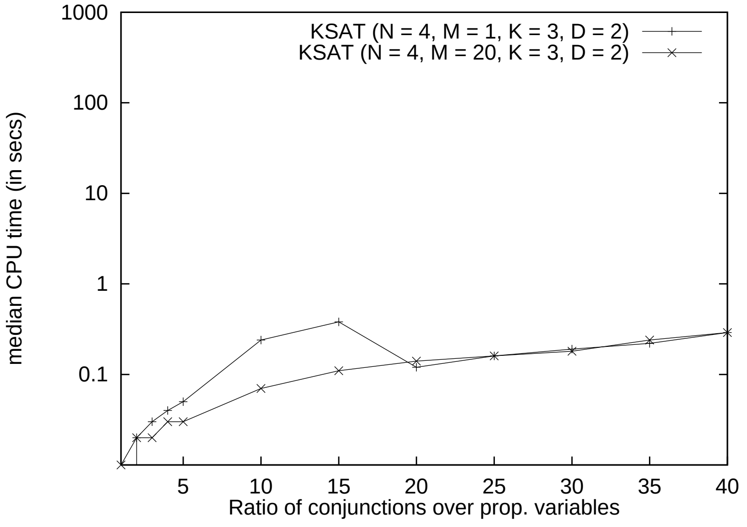

Figure 19: Varying the parameter M for KSAT

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 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. 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 7:
The formula ϕ_6 given by

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

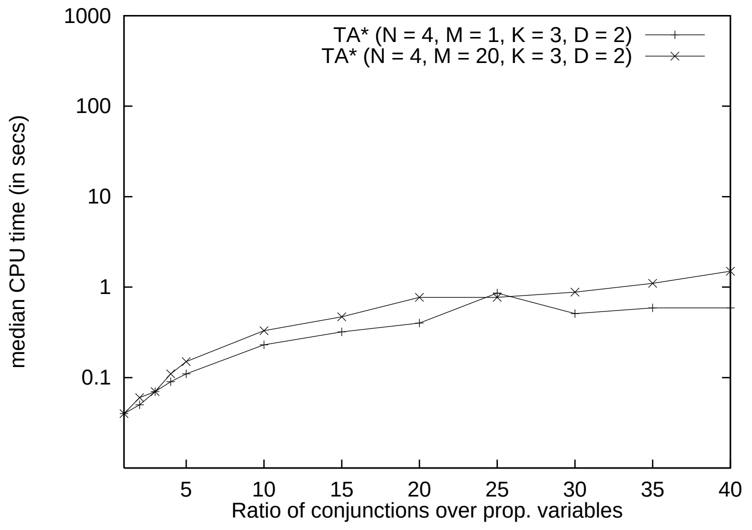

Figure 20: Varying the parameter M for the translation approach

is satisfiable. KSAT will first apply the `dp_unit` rule replacing $\Box_1(q \vee r)$ by $\bot$. The first conjunct of ϕ_6 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_6 true propositionally, but checking the satisfiability of $\neg(q \vee p) \wedge (p \vee q)$ 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 $\neg(q \vee p) \wedge (r \vee q)$ which succeeds.

In contrast consider the formula ϕ_7 given by

$$\begin{array}{l} \quad (\Box_2(p \vee q) \vee \Box_1(r \vee q)) \\ \wedge \neg\Box_1(q \vee p), \end{array}$$

which is like ϕ_6 except the first occurrence of a $\Box_1$ is replaced by $\Box_2$. KSAT proceeds in the same way as for ϕ_6. It replaces $\Box_1(q \vee p)$ 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 $\neg(q \vee p) \wedge (p \vee q)$ we just have to check that $\neg(q \vee p)$ is satisfiable, because $p \vee q$ occurs below a different modality. Since this check succeeds ϕ_7 is satisfiable. Evidently, the computation for ϕ_7 is easier than for ϕ_6.

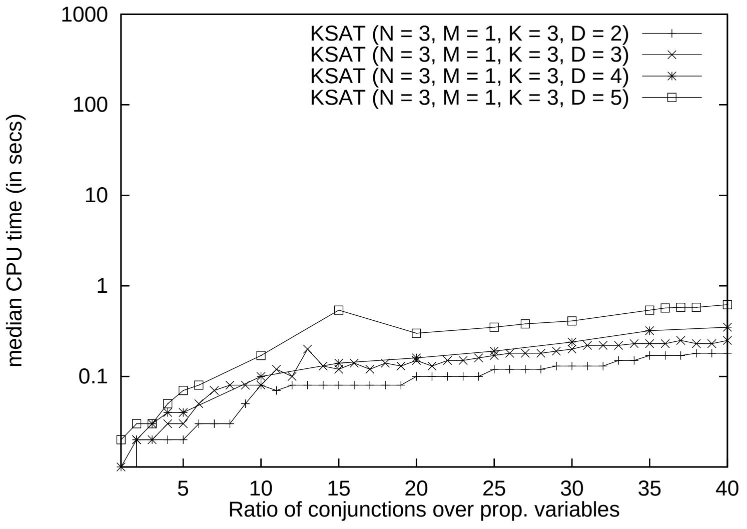

Figure 21: Varying the parameter D for KSAT

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 22 and 21 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 23. 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.

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

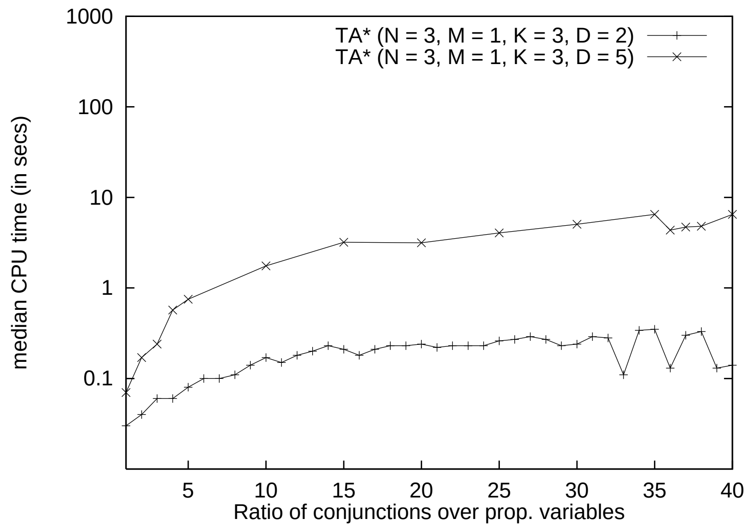

Figure 22: Varying the parameter D for the translation approach

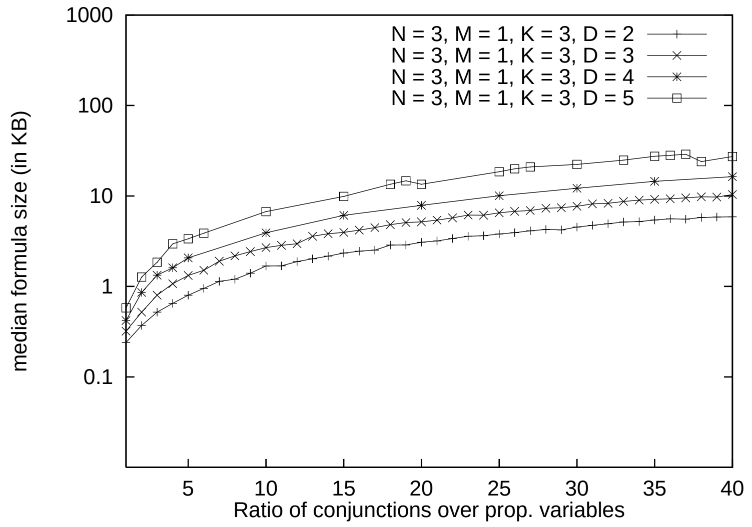

Figure 23: The influence of the parameter D on the formula size

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.

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 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 24 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 26) for 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.

The performances of KSAT, $\mathcal{KRIS}$*, LWB* and the translation approach on the settings **PS12** and **PS13** are given in Figures 25 and 30. Figures 26 to 34 present the corresponding percentile graphs. 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.

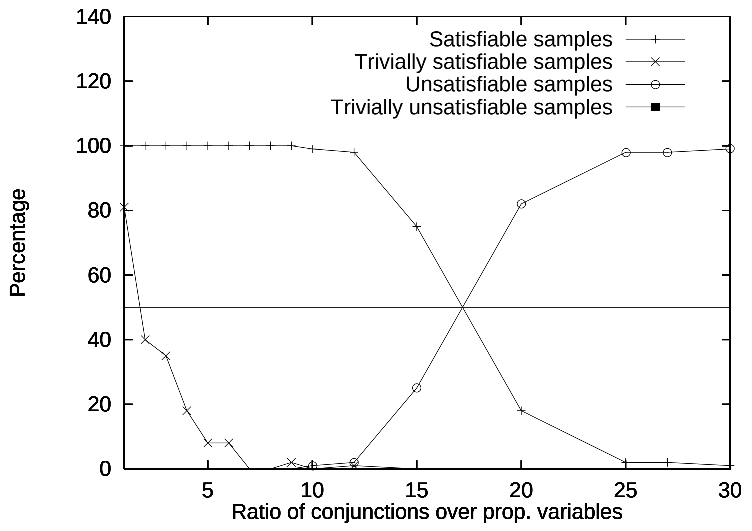

Figure 24: The quality of the test set for **PS12**

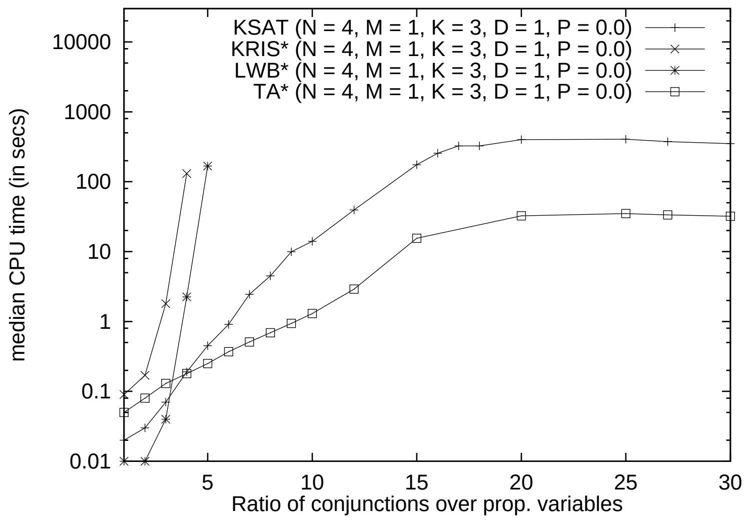

Figure 25: The median performances for **PS12**

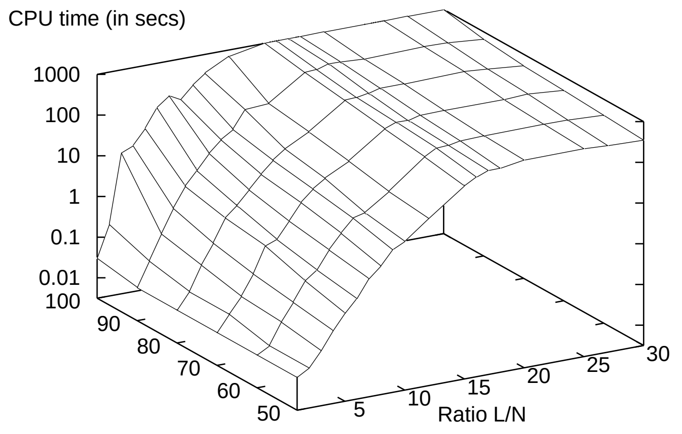

Figure 26: The percentile graphs for KSAT on **PS12**

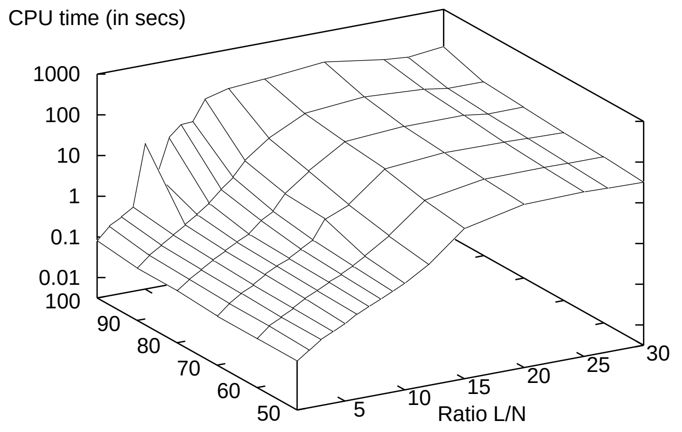

Figure 27: The percentile graphs for the translation approach on **PS12**

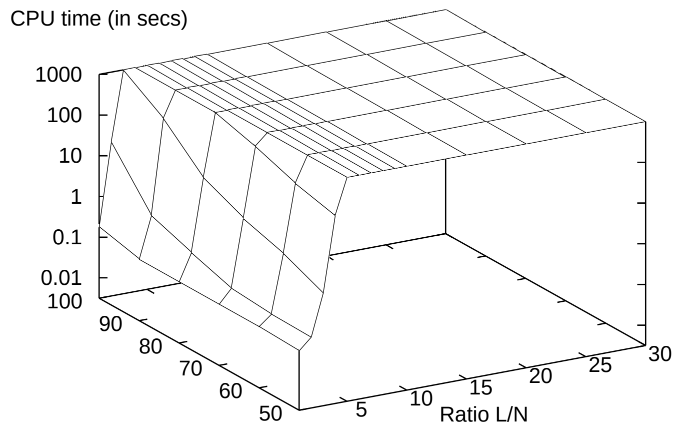

Figure 28: The percentile graphs for $\mathcal{KRIS}^*$ on **PS12**

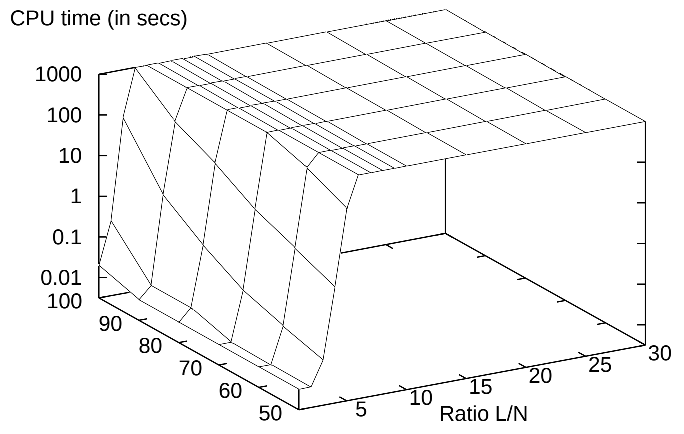

Figure 29: The percentile graphs for the Logics Workbench on **PS12**

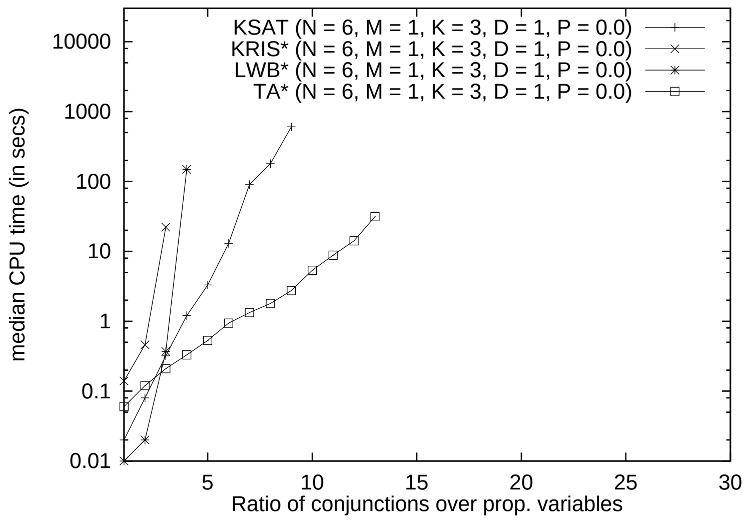

Figure 30: The median performances for **PS13**

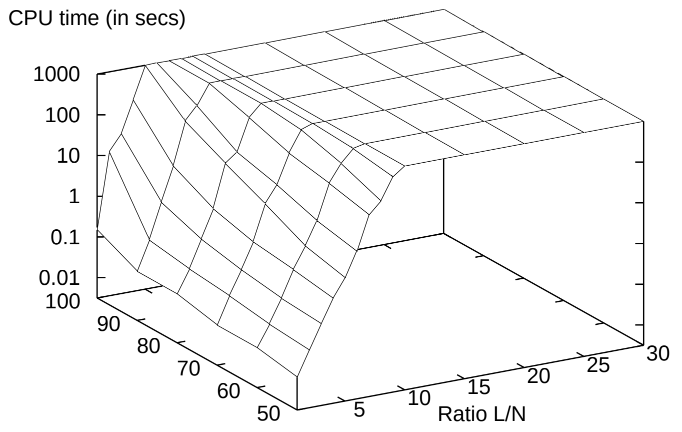

Figure 31: The percentile graphs for Ksat on **PS13**

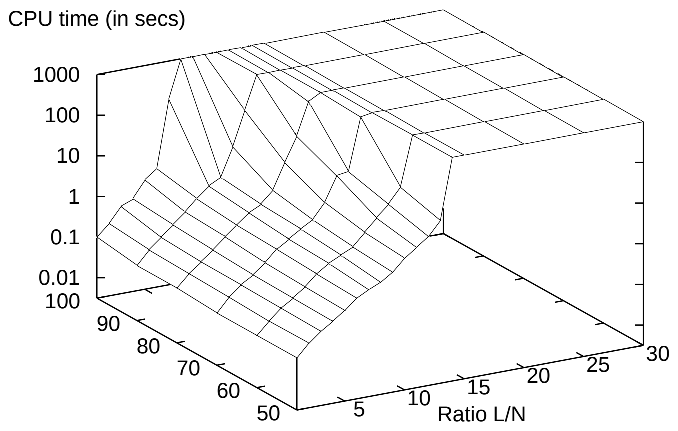

Figure 32: The percentile graphs for the translation approach on **PS13**

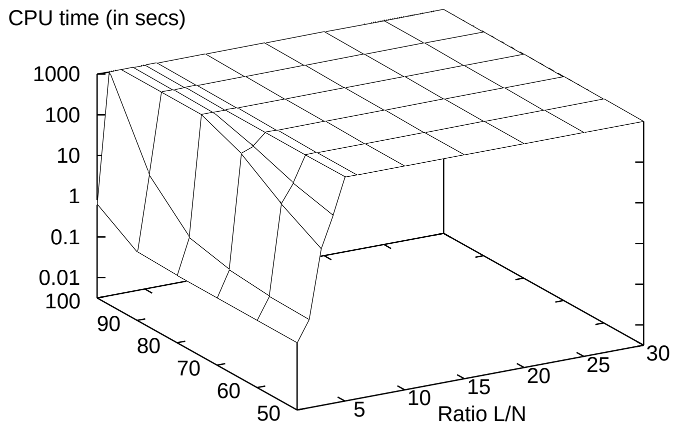

Figure 33: The percentile graphs for $\mathcal{KRIS}^*$ on **PS13**

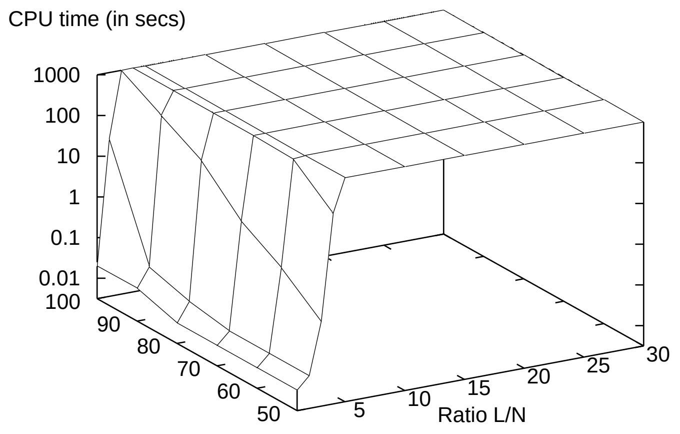

Figure 34: The percentile graphs for the Logics Workbench on **PS13**

10 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 [5]. Our analysis shows the random generator used by Giunchiglia and Sebastiani is not adequate as it distorts the evaluation.

The basic algorithm of KSAT is very similar to the algorithm of $\mathcal{KRIS}$. The essential differences between KSAT and $\mathcal{KRIS}$ 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 $\vee$-$\texttt{simp}_0$ and $\vee$-$\texttt{simp}_1$, which are applied occasionally.

2. KSAT utilizes a heuristic for selecting the particular disjunction for the application of disjunction elimination (namely, `dp_split`). By contrast, $\mathcal{KRIS}$ processes 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.

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 KSAT and $\mathcal{KRIS}$ concerning their performance is the following:

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

2. There is no evidence that the particular heuristics used by KSAT provides an overall improvement of performance. In particular, we have shown that the median CPU time consumption of $\mathcal{KRIS}$* for ratios L/N between 4 and 11 on the parameter setting N=8, M=1, K=3, D=2, P=0.5 is better than that of KSAT. Furthermore, there are no indications that the superior performance of KSAT on samples generated for smaller values of N is due to the particular heuristic used by KSAT.

   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 for any samples.

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, Example 6 shows that in its present form KSAT can not make optimal use of the information provided by a failure of an intermediate call to KM.

Further improvements of 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.

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 formulae. The reason, we believe, is a fundamental difference in method. The translation approach adopts a *global* method, whereas KSAT and tableaux or sequent-based systems adopt a *local* method.

The local methods proceed by constructing a (derivation) tree in a node-by-node manner. The tree grows by the application of the branching rules ($\vee$-`elim`, $(l\vee)$, $(r\wedge)$, and `dp_split`) and the rules for new nodes ($\Diamond_i$-`elim`, $(l\Diamond)$, $(r\Box)$, and KM). The tree is implicit only in the control structure of the local systems. The systems perform a depth-first search through the tree. At each node the systems are doing in essence ordinary propositional tests for satisfiability. Local approaches are based on the internal view of Kripke semantics that reduces truth to truth in worlds (on the element level) and the derivation tree closely resembles the underlying frame.

For example, the internal reading of $\Diamond_1\Diamond_1 p \wedge \Box_1\Box_1\neg p$ is that there is some world at level two that is labeled $p \wedge \neg p$. The local methods cannot directly see this information, but derive it in a node-by-node construction. They start with the root of the tree, followed by the construction of nodes at depth one, until they have constructed the node labeled with $p \wedge \neg p$ at depth two.

The global method of the translation approach transfers the underlying tree structure from the meta level to the object level in the first-order translation of the given modal formula. The theorem prover reasons about the structure of the tree on the object level, and is not restricted to a depth-first search.

For example, the underlying tree structure of the formula $\Diamond_1\Diamond_1 p\wedge\Box_1\Box_1\neg p$ is made explicit in the global presentation of four clauses:

(5) $\quad def_1$

(6) $\quad def_1(\underline{a})$

(7) $\quad P_{11}(\underline{a}, \underline{b})$

(8) $\quad \neg def_1 \vee \neg def_1(x) \vee \neg P_{11}(x, y)$

The theorem prover can directly resolve the clauses (7) and (8) to derive

$$\neg def_1 \vee \neg def_1(\underline{a}).$$

Unit propagation with the first two clauses reveals the unsatisfiability of the formula. Example 6 also illustrates the importance of this form of global reasoning.

It is open which resolution inference rules and search strategies perform best for basic modal logic and its extensions.

References

[1] Franz Baader and Bernhard 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, Kaiserslautern, Germany, July 1–3 1991. Springer.

[2] Leo Bachmair and Harald 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, Kaiserslautern, FRG, July 24–27 1990. Springer.

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

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

[5] Byungki Cha and Kazuo Iwama. Performance test of local search algorithms using new types of random CNF formulas. In Chris S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 304–311, Montréal, Québec, Canada, August 20–25 1995. Morgan Kaufmann.

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

[7] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*, volume 169 of *Synthese Library, Studies in Epistemology, Logic, Methodology, and Philosophy of Science*. D. Reidel Publishing Company, Dordrecht, Holland, 1983.

[8] Fausto Giunchiglia and Roberto Sebastiani. Building decision procedures for modal logics from propositional decision procedures: Case study of modal K. In Michael A. McRobbie and John K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 583–597. Springer-Verlag, 1996.

[9] Fausto Giunchiglia and Roberto Sebastiani. A SAT-based decision procedure for alc. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 304–314, Cambridge, MA, USA, November 5–8 1996.

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

[11] Alain Heuerding, Gerhard Jäger, Stefan Schwendimann, and Michael Seyfried. The Logics Workbench LWB: A snapshot. *Euromath Bulletin*, 2(1):177–186, 1996.

[12] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465, San Jose, CA, USA, July 12–16 1992. MIT Press.

[13] Hans Jürgen Ohlbach. Semantics based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[14] Hans Jürgen Ohlbach. Translation methods for non-classical logics: An overview. Research Report MPI-I-93-225, Max-Planck-Institut für Informatik, Saarbrücken, Germany, June 1993.

[15] Hans Jürgen Ohlbach and Renate A. Schmidt. Functional translation and second-order frame properties of modal logics. Research Report MPI-I-95-2-002, Max-Planck-Institut für Informatik, Saarbrücken, Germany, January 1995.

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

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

[18] Renate A. Schmidt. Resolution is a decision procedure for many propositional modal logics. Research Report MPI-I-97-2-002, Max-Planck-Institut für Informatik, Saarbrücken, Germany, January 1997. The extended abstract version is to appear in M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, editors, *Advances in Modal Logic '96*, CSLI Publications 1997.

[19] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept description with complements. *Artifical Intelligence*, 48:1–26, 1991.

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

[21] Christoph Weidenbach, Bernd Gaede, and Georg Rock. SPASS & FLOTTER version 0.42. In M.A. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 141–145, New Brunswick, NJ, USA, July 30–August 3 1996. Springer. For more recent versions see `http://www.mpi-sb.mpg.de/guide/software/spass.html`.