Scientific Benchmarking with Temporal Logic Decision Procedures

Ullrich Hustadt
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
U.Hustadt@csc.liv.ac.uk

Renate A. Schmidt
Department of Computer Science
University of Manchester
Manchester M13 9PL, UK
schmidt@cs.man.ac.uk

Abstract

In this paper we propose a hypothesis-driven design of the empirical analysis of different decision procedures which we refer to as *scientific benchmarking*. The approach is to start by choosing the benchmark problems for which, on the basis of analytical considerations, we expect a particular decision procedure to exhibit a behaviour different from another decision procedure. Then empirical tests are performed in order to verify the particular hypothesis concerning the decision procedures under consideration. As a case study, we apply this methodology to compare different decision procedures for propositional temporal logic. We define two classes of randomly generated temporal logic formulae which we use to investigate the behaviour of two tableaux-based temporal logic approaches using the Logics Workbench, a third tableaux-based approach using the *STeP* system, and temporal resolution using a new prover called *TRP*.

1 Introduction

In recent years it has become common to compare decision procedures for a wide variety of logics, including Boolean logic, quantified Boolean logic, basic modal logic, and description logics on sets of randomly generated classes of formulae [1, 3, 7, 11, 16, 17, 18, 24, 25, 27]. Although this form of analysis of algorithms and their implementations has lead to a number of interesting results and has certainly contributed to recent progress in the sophistication of decision procedures for the logics under consideration, empirical analysis with randomly generated problem sets is not without problems.

One of the main problems stems from the fact that underlying any class of randomly generated problems is a probability density function μ which assigns probabilities to instances of the problem class and the fact that the analysis is sensitive to the choice of μ. The most famous example illustrating this problem is the density function underlying a result by Goldberg [12, 13]. Goldberg has shown (analytically) that the satisfiability problem of Boolean formulae can be solved by a DPLL procedure in polynomial time in the average case. Subsequently, Franco and Paull [10] proved that this result is due to the density function he assumed and not a particular feature of the DPLL algorithm, and that for a more reasonable density function on the class of Boolean formulae, a variant of the DPLL procedure needs exponential time in the average case.

A related problem is that it is the aim of most comparisons to demonstrate differences between various algorithms, data structures, or implementation techniques. They are often used to illustrate the usefulness of a new idea by comparing an implementation A, incorporating this idea, with an implementation B that does not. However, even if we consider a problem class together with a probability density function μ that makes the problem class reasonably hard in the average case, there is no guarantee that we will be able to see any differences in the average performance of implementations A and B. It may well be that the particular problems on which A is superior to B have a low probability with respect to μ and can therefore not influence the average performance sufficiently to have a noticeable impact. This might be justified if μ models the probability of problems close to what we expect in the 'real world', but this seems to be rarely the case for the logics mentioned in the beginning and the density functions described in the literature.

The problem is aggravated if we consider implementations A and B of decision procedures which vary in more aspects than one. In this case, even if we can

observe a noticeable difference in the average performance, we have to explain which particular aspects do contribute to this difference and which do not. Again, the choice of the density function μ can influence our ability to provide convincing evidence for any such explanation.

Hooker [15] proposes the following experimental design to overcome some of the problems related to this kind of empirical analysis: We should start by setting up a list of hypotheses about factors that could affect the performance of an algorithm. Some of these factors can be related to features of the problems we like to deal with, e.g. size of the problems, a specific structure of the problems, etc. Other factors can be related to the algorithm itself, e.g. its inference rules, its heuristics, its redundancy checks, etc. When formulating a hypothesis we should use some abstract measure to describe the effect of a factor, for example, the number of nodes in a tableau or the length of a refutation using resolution, instead of simply relying on running time. For each factor we should set up a benchmark suite suitable for verifying our hypothesis about the influence of the factor on the performance of the algorithm. The benchmark problems can be purely artificial and need not be related to any 'real world' problems. So randomly generated problems are well suited for this purpose. Of course, we still have to be careful that the generated problems possess the problem characteristics needed for the test. Since the intention is not to perform a competitive test, it is not necessary to have an efficient implementation of our algorithm at hand. The only necessary prerequisite is that we are able to alter the implementation to test our hypotheses about the algorithm itself, for example, we need to be able to turn off specific inference rules and adjust specific heuristics. Hooker calls this kind of experimental design *scientific testing*. It is an implicit assumption of the scientific testing methodology that we restrict ourselves to variations of one basic algorithm.

Of course, this is a considerable restrictions if we are interested in assessing a broad variety of decision procedures for a particular logic. Simulation results relating, for example, derivations in tableau calculi for basic modal logic with those in the translation approach [4, 19, 20] or derivations using the inverse method with the automata approach to modal satisfiability [2], may still form a basis for an abstract measure for the performance of decision procedures based on different calculi. However, if this abstract measure is on the level of inference steps in the calculus underlying each of the decision procedures, then one may argue that this puts certain approaches at a disadvantage, in particular, if they allow the use of specialised data structures and algorithms which result in a much higher inference rate than for competing approaches. In such a case it may also be important that the implementation of the decision procedures under consideration is sufficiently sophisticated to enable each approach to show its full strength. Consequently, we will in most cases have to rely on existing implementations, since otherwise the implementational effort required will be excessive. So, in the absence of an abstract measure relating the performance of different decision procedures, or if we feel that such an abstract measure would lead to an incorrect assessment, measuring the running time of existing implementations of decision procedures for a particular logic can still be a viable option. What remains of the scientific testing methodology is that the benchmark problems are chosen to verify a particular hypothesis concerning the decision procedures under consideration. We call this kind of experimental design *scientific benchmarking* or *hypothesis-driven benchmarking*.

There is a danger that scientific benchmarking might be abused to highlight only the strengths of a particular decision procedure over others. This is not the intention. On the contrary, we believe the aim should be to identify and study both the strengths and weaknesses of decision procedures, to gain insight, and therefore, increase the scientific value of a study. Thus, an important component of scientific benchmarking is the analytical analysis of a given problem class with respect to the features of, and/or methods used in, the decision procedures which are subject of the analysis.

In the following we apply this scientific benchmarking approach to a comparison of decision procedures for propositional linear time logic PLTL. Our aim is to identify some problem sets with characteristic properties for which we expect a particular decision procedure to exhibit a behaviour different from other decision procedures.

For propositional linear time logic PLTL there are three major approaches: (i) tableaux- and sequent-based calculi, (ii) automata-based approaches, and (iii) temporal resolution calculi. In most publications on tableaux-based calculi for PLTL [14, 29] the decision procedure proceeds in two phases: Given a PLTL formula φ which has to be tested for satisfiability, first, the procedure creates a pre-model $\mathcal{M}$ for the formula by applying tableaux expansion rules to φ, then, in the second phase the procedure checks whether $\mathcal{M}$ satisfies all eventuality formulae. Two decision procedures based on this approach have been developed by Janssen [21] and McGuire et al. [22]. A C++ implementation of Janssen's procedure is incorporated

as the 'satisfiability' function in the PLTL module of the Logics Workbench Version 1.1, while the *STeP* system [23] includes an implementation of McGuire's procedure.

In contrast, Schwendimann [26] presents a one-phase tableau calculus which checks locally, on-the-fly, for the fulfilment of eventuality formulae on a branch-by-branch basis. A C++ implementation of this calculus is incorporated as the 'model' function in the PLTL module of the Logics Workbench Version 1.1.

Finally, the resolution method for PLTL proposed by Fisher [8] (see also [5, 6, 9]) involves the translation of PLTL formulae to a clausal normal form, classical resolution within states and temporal resolution over states between eventuality formulae like $\Diamond \neg p$ and formulae that together imply $\Box p$. We have used an experimental implementation, called *TRP* (written in SICStus Prolog 3.8.6), for our experiments. *TRP* is an extension of a first-order resolution theorem prover by temporal resolution.

Since the resolution method for PLTL is less well-established compared to the tableaux-based (and automata-based) approaches, we will focus on highlighting some strengths and weaknesses of this approach compared to the three tableaux-based decision procedures mentioned above. In particular, we will design two classes of randomly generated PLTL formulae, the first class will demonstrate that *TRP* can exhibit better performance than the other approaches, while the second class will demonstrate a shortcoming of *TRP* that allows one of the other approaches to outperform *TRP*.

2 Basics of PLTL

Let P be a set of propositional variables. The set of formulae of *propositional linear time logic* PLTL (over P) is inductively defined as follows: (i) $\top$ is a formula of PLTL, (ii) every propositional variable of P is a formula of PLTL, (iii) if φ and ψ are formulae of PLTL, then $\neg\varphi$ and $(\varphi \vee \psi)$ are formulae of PLTL, and (iv) if φ and ψ are formulae of PLTL, then $\bigcirc\varphi$ (in the next moment of time φ is true), $\Diamond\varphi$ (sometimes in the future φ is true), $\Box\varphi$ (always in the future φ is true), $\varphi\,\mathcal{U}\,\psi$ (φ is true until ψ is true), and $\varphi\,\mathcal{W}\,\psi$ (φ is true unless ψ is true) are formulae of PLTL. Other Boolean connectives including $\bot$, $\wedge$, $\rightarrow$, and $\leftrightarrow$ are defined using $\top$, $\neg$, and $\vee$. A formula of the form $\Diamond\varphi$ is called a $\Diamond$-*formula*. Analogously, for the remaining temporal operators.

We also use $\top$ and $\bot$ to denote the empty conjunction and disjunction, respectively. A *literal* is either p or $\neg p$ where p is a propositional variable.

PLTL-formulae are interpreted over ordered pairs $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ where (i) $\mathcal{S}$ is an infinite sequence of *states* $(s_i)_{i\in\mathbb{N}}$ and (ii) ι is an *interpretation function* assigning to each state a subset of P.

For an interpretation ι and a state $s_i \in \mathcal{S}$ we can define a *propositional valuation* I by $I(p) = (p \in \iota(s_i))$ for every $p \in \mathsf{P}$. On the other hand, given an infinite sequence $(I_i)_{i\in\mathbb{N}}$ of propositional valuations, we can define an interpretation function ι by $\iota(s_i) = \{p \mid I_i(p) = \text{true}\}$.

We define a binary relation $\models$ between a PLTL-formula φ and a pair consisting of a PLTL-interpretation $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ and a state $s_i \in \mathcal{S}$ as follows.

$\mathcal{I}, s_i \models \mathsf{true}$
$\mathcal{I}, s_i \models p$ iff $p \in \iota(s_i)$
$\mathcal{I}, s_i \models \neg\varphi$ iff $\mathcal{I}, s_i \not\models \varphi$
$\mathcal{I}, s_i \models \varphi \wedge \psi$ iff $\mathcal{I}, s_i \models \varphi$ and $\mathcal{I}, s_i \models \psi$
$\mathcal{I}, s_i \models \varphi \vee \psi$ iff $\mathcal{I}, s_i \models \varphi$ or $\mathcal{I}, s_i \models \psi$
$\mathcal{I}, s_i \models \bigcirc\varphi$ iff $\mathcal{I}, s_{i+1} \models \varphi$
$\mathcal{I}, s_i \models \Box\varphi$ iff for all $j \in \mathbb{N}$, $j \geq i$ implies $\mathcal{I}, s_j \models \varphi$
$\mathcal{I}, s_i \models \Diamond\varphi$ iff there exists $j \in \mathbb{N}$ such that $j \geq i$ and $\mathcal{I}, s_j \models \varphi$
$\mathcal{I}, s_i \models \varphi\,\mathcal{U}\,\psi$ iff there exists $j \in \mathbb{N}$ such that $j \geq i$, $\mathcal{I}, s_j \models \psi$, and for all $k \in \mathbb{N}$, $j > k \leq i$ implies $\mathcal{I}, s_k \models \varphi$
$\mathcal{I}, s_i \models \varphi\,\mathcal{W}\,\psi$ iff $\mathcal{I}, s_i \models \varphi\,\mathcal{U}\,\psi$ or $\mathcal{I}, s_i \models \Box\varphi$

If $\mathcal{I}, s_i \models \varphi$ then we say φ is *true*, or *holds*, at s_i in $\mathcal{I}$. An interpretation $\mathcal{I}$ *satisfies* a formula φ iff φ holds at s_0 in $\mathcal{I}$ and it *satisfies* a set N of formulae iff for every formula $\psi \in N$, $\mathcal{I}$ satisfies ψ. In this case, $\mathcal{I}$ is a *model* for φ and N, respectively, and we say φ and N are (PLTL-)satisfiable. The satisfiability problem of PLTL is known to be PSPACE-complete [28].

Arbitrary PLTL-formulae can be transformed into a *separated normal form* (SNF) in a satisfiability equivalence preserving way using a renaming technique replacing non-atomic subformulae by new propositions and removing all occurrences of the $\mathcal{U}$ and $\mathcal{W}$ operator [8, 9]. The result is a set of *SNF clauses* of the following form (which differs slightly from [8, 9]).

(initial clause) $\bigvee_{i=1}^{n} L_i$
(global clause) $\Box(\bigvee_{j=1}^{m} K_j \vee \bigvee_{i=1}^{n} \bigcirc L_i)$
(sometime clause) $\Box(\bigvee_{j=1}^{m} K_j \vee \Diamond L)$

Here, K_j, L_i, and L (with $1 \leq j \leq m$, $1 \leq m$, and $1 \leq i \leq n$, $0 \leq n$) denote literals.

Given a PLTL-formula φ the transformation to SNF starts with a set $\{q, \neg q \vee \varphi\}$, where q is a new propositional variable, and exhaustively applies a set of transformation rules to this set. We do not give a complete description, but only present the transformation rules dealing with formulae of the form $\Box\,\varphi$ and $\varphi\ \mathcal{W}\ \psi$ which are important for the understanding of the temporal resolution rule presented below.

$$\{\neg p_1 \vee \Box\,\varphi\} \Rightarrow \left\{\begin{array}{l}\neg p_1 \vee p_2 \\ \Box(\neg p_2 \vee \bigcirc p_2) \\ \Box(\neg p_2 \vee \varphi)\end{array}\right\}$$

if φ is a literal and p_2 is new.

$$\{\neg p_1 \vee \varphi\ \mathcal{W}\ \psi\} \Rightarrow \left\{\begin{array}{l}\neg p_1 \vee \varphi \vee \psi \\ \neg p_1 \vee p_2 \vee \psi \\ \Box\,\neg p_2 \vee \bigcirc(\varphi \vee \psi) \\ \Box\,\neg p_2 \vee \bigcirc(p_2 \vee \psi)\end{array}\right\}$$

if φ and ψ are literals and p_2 is new. The result of the transformation of a formula φ to separated normal form is denoted by $\mathrm{snf}(\varphi)$.

We need some additional terminology for SNF clauses. If $C = \Box(\bigvee_{j=1}^{m} K_j \vee \bigvee_{i=1}^{n} \bigcirc L_i)$ is a global clause, then by $\mathrm{now}(C)$ we denote $\bigvee_{j=1}^{m} K_j$ and by $\mathrm{next}(C)$ we denote $\bigvee_{i=1}^{n} L_i$. If $C = \Box(\bigvee_{i=1}^{n} L_i)$, then $\mathrm{now}(C)$ will denote the empty clause and $\mathrm{next}(C)$ will denote $\bigvee_{i=1}^{n} L_i$.

Lemma 1 ([8, 9]). *Let φ be a formula of* PLTL*. Then, φ is* PLTL*-satisfiable if and only if* $\mathrm{snf}(\varphi)$ *is* PLTL*-satisfiable, and* $\mathrm{snf}(\varphi)$ *is computable in time polynomial in the size of φ.*

3 Random PLTL-formulae

The classes we introduce are intended to test the following hypothesis: We conjecture that the temporal resolution calculus will perform better than the three tableau calculi under consideration on problem classes consisting of sets of SNF clauses which impose uniform non-trivial constraints on the states of potential models. On the other hand, we expect the tableau calculi to have an advantage over temporal resolution if a considerable number of clauses impose only trivial constraints on most of the states, particularly, if these clauses include sometime clauses.

The intuition underlying the notion of a clause imposing a trivial or non-trivial constraint on the states is best explained with examples. Consider the PLTL-formula

$$\varphi_1 = \neg p \wedge \Box(\neg p \vee \bigcirc(p \wedge \psi))$$

where ψ is an arbitrary, large PLTL-formula. Suppose we want to construct a model $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ for φ_1. The first conjunct of φ_1 imposes the constraint that p has to be false in the initial state s_0, that is, $p \notin \iota(s_0)$. But, then the second conjunct of φ_1 imposes no further constraints on s_0. Since $\neg p$ is true at s_0, so is $\neg p \vee \bigcirc(p \wedge \psi)$. The construction of the model can then be completed by defining $\iota(s_{i+1}) = \iota(s_0)$ for all $i \in \mathbb{N}$. In this construction the truth of ψ in any of the states of $\mathcal{I}$ was never considered. Thus, although ψ is a large PLTL-formula by assumption and although it may seem at first sight that the second conjunct of φ_1 imposes a complex constraint on potential models of φ_1, it turns out that it imposes only a 'trivial' constraint since we were able to satisfy the constraint by simply making p false at every state.

Now, consider the PLTL-formula

$$\varphi_2 = \Box(p \vee q) \wedge \Box(\neg p \vee \bigcirc \psi_1) \wedge \Box(\neg q \vee \bigcirc \psi_2)$$

where ψ_1 and ψ_2 are arbitrary, large PLTL-formulae. Again, we want to construct a model $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ for this formula. In constrast to φ_1, where the first conjunct only imposed a constraint on the first state of the model, here the first conjunct $\Box(p \vee q)$ imposes a uniform constraint on all states of $\mathcal{I}$. If we start the construction at state s_0, then we have to make either p or q true at this state. We can also see that, depending on our choice, we will have to consider the truth of $\bigcirc\psi_1$ or $\bigcirc\psi_2$ which, given that ψ_1 and ψ_2 are large formulae, can turn out to be non-trivial. Furthermore, since $\bigcirc\psi_1$ and $\bigcirc\psi_2$ impose constraints on the next state s_1, which we have not considered yet, we have no indication whether making p true at s_0 will lead to an 'easier' model construction than making q true at s_0 or vice versa. In addition, $\Box(p \vee q)$ enforces that $p \vee q$ has to be true at s_1 as well; a constraint that may interact with ψ_1 or ψ_2 being true at s_1. Thus, in the context of the second and third conjunct of φ, the first conjunct $\Box(p \vee q)$ imposes a non-trivial constraint on our model construction.

Since tableau derivations are closely related to model construction approaches as the one we have used for the two example formulae, it is reasonable to expect that tableau calculi find formulae similar to φ_1 easier than formulae similar to φ_2. Before we explain why we expect that temporal resolution will behave in the opposite way, we first define the classes of formulae with which we are going to test the hypothesis.

Based on Lemma 1 it is clear that in the construction of classes of formulae suitable to test the hypothesis we can restrict ourselves to sets of SNF clauses instead of arbitrary PLTL-formulae. The similarity of

SNF clauses to propositional clauses makes it easy to generalise the standard random generator for uniform propositional kSAT clauses to SNF clauses. This can be done in a number of ways. The random generator we have used in our experiments takes as input the number of propositional variables n, the number of clauses l, a probability p, and a number k.

For the first class of PLTL-formulae, denoted by $\mathcal{C}^1_{ran}$, which will test the first part of the hypothesis, each randomly generated formula φ is a conjunction of SNF clauses of the following form

$$\begin{aligned}
&\Box(\bigcirc L^1_1 \vee \ldots \vee \bigcirc L^1_k) \wedge \ldots \wedge \Box(\bigcirc L^l_1 \vee \ldots \vee \bigcirc L^l_k)\\
&\wedge \Box(\neg p_1 \vee \Diamond p_2)\\
&\wedge \Box(\neg p_2 \vee \Diamond p_3)\\
&\vdots\\
&\wedge \Box(\neg p_n \vee \Diamond p_1),
\end{aligned}$$

where for each global clause, the literals $L^i_1, \ldots, L^i_k$ are generated by choosing k distinct variables randomly from the set $\{p_1, \ldots, p_n\}$ of n propositional variables and by determining the polarity of each literal with probability p. The sometime clauses included in φ only depend on the parameter n.

For the second class of PLTL-formulae, denoted by $\mathcal{C}^2_{ran}$, which will test the second part of the hypothesis, each randomly generated formula φ is a conjunction of initial, global and sometime clauses of the following form.

$$\begin{aligned}
&(r_1 \vee L^1_1 \vee \ldots \vee L^1_k) \wedge \ldots \wedge (r_1 \vee L^l_1 \vee \ldots \vee L^l_k)\\
&\wedge \Box(\neg r_n \vee \bigcirc r_1)\\
&\wedge \Box(\neg r_{n-1} \vee \bigcirc r_n)\\
&\vdots\\
&\wedge \Box(\neg r_1 \vee \bigcirc r_2)\\
&\wedge \Box(\neg r_n \vee \bigcirc \neg q_n) \wedge \ldots \wedge \Box(\neg r_1 \vee \bigcirc \neg q_n)\\
&\wedge (\neg r_1 \vee q_1) \wedge (\neg r_1 \vee \neg q_n)\\
&\wedge \Box(\neg q_1 \vee \Diamond s_2) \wedge \Box(\neg s_2 \vee q_2 \vee \bigcirc q_n \vee \ldots \vee \bigcirc q_3)\\
&\vdots\\
&\wedge \Box(\neg q_{n-1} \vee \Diamond s_n) \wedge \Box(\neg s_n \vee q_n)
\end{aligned}$$

For each of the first l initial clauses, the literals $L^i_1, \ldots, L^i_k$ are generated by choosing k distinct variables randomly from the set $\{p_1, \ldots, p_n\}$ of n propositional variables and by determining the polarity of each literal with probability p. The global and sometime clauses included in φ only depend on the parameter n.

In all our experiments, for both classes, the parameters k and p were fixed to 3 and 0.5, respectively.

The restricted form of global and sometime clauses in both classes indicates that they form strict subclasses of the class of sets (or conjunctions) of SNF clauses. Thus, not every PLTL-formula is equivalent to some formula in $\mathcal{C}^1_{ran}$or $\mathcal{C}^2_{ran}$. It turns out that the satisfiability problem of both $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ has reduced complexity.

Theorem 2. *The satisfiability problem for $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ is in* NP.

Proof sketch. Let φ be a SNF formula in $\mathcal{C}^1_{ran}$ over n propositional variables with global clauses $C_1, \ldots, C_l$. Let ψ' be the propositional formula $\text{next}(C_1) \wedge \ldots \wedge \text{next}(C_l)$. Let ψ_0 be the propositional formula $\psi' \wedge \neg p_0 \wedge \ldots \wedge \neg p_n$, and let ψ_i be the propositional formula $\psi' \wedge p_i$ for every i, $1 \leq i \leq n$. Then φ is satisfiable iff ψ_0 is satisfiable or for every i, ψ_i is satisfiable. Since each ψ_i, $0 \leq i \leq n$, is a propositional formula of length smaller than that of φ, these $n+1$ satisfiability test can be performed by a non-deterministic Turing machine in time polynomial in the length of φ.

Let φ be a SNF formula in $\mathcal{C}^2_{ran}$ over n propositional variables with global and sometime clauses $C_1, \ldots, C_{4n-2}$, and initial clauses $I_1, \ldots, I_{l+2}$. For an initial clause I_j let $I_j[r_1/\bot]$ denote the clause obtained from I_j by substituting $\bot$ (falsum) for r_1. It is straightforward to check that $G = r_1 \wedge \bigwedge_{1 \leq i \leq 4n-2} C_i$ is unsatisfiable. As far as the initial clauses are concerned, we see that almost all of them contain r_1 as a positive literal. Obviously, if $I = \bigwedge_{1 \leq i \leq l+2} I_i[r_1/\bot]$ is satisfiable, we can build a PLTL model $\mathcal{I}$ for φ where r_1 is false in the initial state s_0 and which satisfies all clauses of φ. However, if I is unsatisfiable, then we can only satisfy the set of initial clauses by assuming that r_1 is true in s_0. But then we will not be able to satisfy G. Thus, φ is unsatisfiable. So, φ is satisfiable iff I is satisfiable. Since I is a propositional formula of length smaller than φ, it follows that the satisfiability problem for $\mathcal{C}^2_{ran}$ is in NP. □

Underlying this theorem are considerations about the satisfiability equivalence of formulae and (un)satisfiability of particular subformulae. It should be stressed that the fact that a formula φ can be transformed into an 'easy' satisfiability equivalent formula φ' does not indicate that φ will be easy for a particular theorem prover. We can only do so if we know that a particular theorem prover actually performs the described transformations. However, the proof of Theorem 2 is the key to understanding the behaviour of the decision procedures under consideration.

To understand why $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ are suitable to test the first and second part of our hypothesis, respectively, we have to take a closer look at *TRP* and the

Initial resolution rules: $\dfrac{C_1 \vee L \qquad \sim L \vee C_2}{C_1 \vee C_2}$ $\qquad$ $\dfrac{C_1 \vee L \qquad \Box(\sim L \vee C_2)}{C_1 \vee C_2}$

where $C_1 \vee L$ and $\sim L \vee C_2$ are initial clauses, $\Box(\sim L \vee C_2)$ is a global clause and $\sim L \vee C_2$ is a propositional clause.

Step resolution rules: $\dfrac{\Box(C_1 \vee L) \qquad \Box(\sim L \vee C_2)}{\Box(C_1 \vee C_2)}$

$$\frac{\Box(C_1 \vee L) \qquad \Box(\bigcirc \sim L \vee D_2)}{\Box(C_1 \vee D_2)} \qquad \frac{\Box(D_1 \vee \bigcirc L) \qquad \Box(\bigcirc \sim L \vee D_2)}{\Box(D_1 \vee D_2)}$$

where $\Box(C_1 \vee L)$ and $\Box(\sim L \vee C_2)$ are global clauses and $C_1 \vee L$ and $\sim L \vee C_2$ are propositional clauses, and $\Box(\bigcirc \sim L \vee D_2)$ and $\Box(D_1 \vee \bigcirc L)$ are global clauses.

Temporal resolution rule:
$$\frac{\begin{array}{cc}\Box(C_1^1 \vee \bigvee_{l=1}^{k_1^1} \bigcirc D_{1,l}^1) & \Box(C_1^n \vee \bigvee_{l=1}^{k_1^n} \bigcirc D_{1,l}^n) \\ \vdots & \vdots \\ \Box(C_{m_1}^1 \vee \bigvee_{l=1}^{k_{m_1}^1} \bigcirc D_{m_1,l}^1) \cdots & \Box(C_{m_n}^n \vee \bigvee_{l=1}^{k_{m_n}^n} \bigcirc D_{m_n,l}^n) \quad \Box(C \vee \Diamond L)\end{array}}{\Box(C \vee (\sim(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C_j^i)\ \mathcal{W}\ L))}$$

where for all i, $1 \leq i \leq n$, $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k_j^i} D_{j,l}^i) \rightarrow \sim L$ and $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k_j^i} D_{j,l}^i) \rightarrow (\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C_j^i)$ are provable.

Figure 1: The temporal resolution calculus

temporal resolution calculus. Besides the standard resolution rule for propositional clause logic (which can only be applied to initial clauses), the temporal resolution calculus contains two inference rules, *step resolution* and *temporal resolution*; see Figure 1. While the step resolution rule resolves two global clauses C_1 and C_2 on complementary literals in $\mathrm{next}(C_1)$ and $\mathrm{next}(C_2)$, the temporal resolution rule resolves a set of global clauses with a sometime clause to derive a set of initial and global clauses. Determining a set of global clauses which can be successfully resolved with a sometime clause is a non-trivial problem, and our implementation follows the approach described in [6]. In *TRP* propositional and step resolution take precedence over temporal resolution. That is, *TRP* will only try to apply the temporal resolution rule when no further applications of the other two rules are possible.

The class $\mathcal{C}_{ran}^1$ has been constructed in such a way that for an unsatisfiable formula φ in $\mathcal{C}_{ran}^1$ either the conjunction of global clauses in φ is already unsatisfiable, which can be detected without any application of the temporal resolution rule, or a small number of comparatively easy applications of the temporal resolution rule will allow us to derive a contradiction. Basically, this is due to the uniform character of the global clauses in φ which impose the same constraints on all states in a potential PLTL model.

In contrast, the tableaux-based decision procedures will try to construct a model for φ. Let $C_1, \ldots, C_l$ be the global clauses in φ. First note, that $C_i = \Box(\bigcirc L_1^i \vee \bigcirc L_2^i \vee \bigcirc L_3^i)$ is equivalent to $\bigcirc \Box(L_1^i \vee L_2^i \vee L_3^i)$. Thus, C_i puts no constraint on the state s_0 in a potential model $\mathcal{I}$ for φ, and enforces the constraint $L_1^i \vee L_2^i \vee L_3^i = \mathrm{next}(C_i)$ on the remaining states s_j, $j > 0$, of $\mathcal{I}$. In analogy to *TRP*, a tableaux-based procedure may easily detect that no model for φ exists if $C_\Box = \bigwedge_{1 \leq i \leq l} \mathrm{next}(C_i)$ is unsatisfiable. This will depend on the presence of sufficiently strong simplification methods for Boolean formulae in the decision procedure. However, if $C_\Box$ is satisfiable, the tableaux-based procedure will construct a set or sequence of states, each with its own valuation for the propositional variables $p_1, \ldots, p_n$, and each satisfying $C_\Box$. In the worst case the procedure constructs 2^n distinct states before this part of the model construction is finished. Note that according to the proof of Theorem 2 a model with at most n states exists for each formula in $\mathcal{C}_{ran}^1$. The procedures will only detect that the interpretation they have constructed is not a model for φ once they have checked whether all the sometime formulae $\Diamond p_i$ contained in the sometime clauses of φ can be satisfied (this is called the *eventuality check*). In the case of *STeP*, this is done in a separate phase, which follows the model construction phase. In this model construction phase *STeP* effectively builds a representation of not just one, but all models of the formula under consideration. This means *STeP* should be very sensitive to the total number of models and consequently we expect it to be the worst performing system on $\mathcal{C}_{ran}^1$. In the case of Schwendimann's one-phase tableau calculus only one interpretation is constructed

at a time and the eventuality check is performed during the search for a model, which should result in better computational behaviour as compared to *STeP*. However, the Schwendimann algorithm will then backtrack and search for an alternative interpretation satisfying φ and only after all alternatives are exhausted will the algorithm conclude that φ is unsatisfiable.

Now consider a satisfiable formula φ in $\mathcal{C}^2_{ran}$ over n propositional variables with global and sometime clauses $C_1, \ldots, C_{4n-2}$, and initial clauses $I_1, \ldots, I_{l+2}$. According to the proof of Theorem 2, φ can only be satisfiable if $I = \bigwedge_{1 \leq i \leq l+2} I_i[r_1/\bot]$ is satisfiable. The initial clauses are the first part of φ considered by any of the tableaux-based decision procedures. Assuming that r_1 is set to false in the initial state, it will be an easy task for the procedure to find a satisfying valuation for the propositional variables $p_1, \ldots, p_n$ occurring in I, the remaining clauses of φ can be satisfying by setting all s_i and q_i, $1 \leq i \leq n$ to false for all states including the initial state. Thus, assuming that the tableaux-based decision procedure searches for a model in the way outlined, it will easily find the model we have just described. The temporal resolution procedure *TRP* on the other hand will always consider the global and sometime clauses of φ regardless as to whether r_1 can be assumed to be false or not. The global and sometime clauses have been chosen in such a way that a successful application of the temporal resolution is possible to each of the sometime clauses, however, only if performed in the correct ordering, starting with $\Box(\neg q_{n-1} \vee \Diamond s_n)$ and ending with $\Box(\neg q_1 \vee s_2)$. For successful and unsuccessful applications of the temporal resolution rule, the global clauses make sure that each application is non-trivial and takes considerable time to perform. After completing all successful applications of the temporal resolution rule, *TRP* will derive $\Box\neg r_1$, which in turn can be used to remove all positive occurrences of r_1 from the initial clauses. However, the simplified initial clauses remain satisfiable. Thus considerable computational resources have been invested into the exploration of a part of φ which a tableaux-based procedure was able to ignore.

What remains to be done, is to verify by empirical experiment that this theoretical analysis correctly reflects the actual behaviour of the procedures under consideration.

4 Experiments

We will now see how *TRP*, *STeP*, and the two tableau calculi incorporated in the Logics Workbench 1.1 perform on $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$.

The tests have been performed on a PC with a 1.3GHz AMD Athlon processor, 512MB main memory, and 1GB virtual memory running RedHat Linux 7.0. For each individual satisfiability test a time-limit of 1000 CPU seconds was used.

Concerning the satisfiability of random formulae in $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ we observe that similar to random kSAT formulae, a random SNF formula is likely to be satisfiable if the number l of clauses is small and it is likely to be unsatisfiable if the number l of clauses is large. Figure 2 shows the percentage of satisfiable random formulae in both classes for $n = 5$ and $n = 12$. To obtain these graphs, for each of the values of l between 1 and $8n$ we have generated a test set of 100 random SNF formulae which were tested for satisfiability. In the case of $\mathcal{C}^1_{ran}$ we see that for ratios l/n smaller than 2, a random formula is most likely satisfiable while for ratios l/n greater than 4, it is most likely unsatisfiable. For ratios l/n between 2 and 4 we note a phase transition in the satisfiability of random SNF formula which becomes more abrupt with greater values of n. For $\mathcal{C}^2_{ran}$, a random formula is most likely satisfiable for ratios l/n smaller than 3.5 and most likely unsatisfiable for ratios l/n greater than 6.5. These graphs have been obtained using the reduction of the satisfiability problem of $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ to propositional logic described in the proof of Theorem 2. The tests have been performed using the propositional logic module of the Logics Workbench 1.1. None of test required more than 80 milliseconds.

Figure 3 shows the median CPU time graphs of all four procedures on $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ for $n = 5$ and $n = 12$. In each graph the vertical line indicates the ratio l/n at which test sets contain 50% satisfiable and 50% unsatisfiable formulae. We observe that on $\mathcal{C}^1_{ran}$, *TRP* performs better than any of the tableaux-based decision procedures except for certain ratios where it is outperformed by Janssen's algorithm (the 'satisfiable' function of the Logics Workbench 1.1).

On $\mathcal{C}^1_{ran}$ for $n = 12$, Janssen's algorithm behaves roughly as predicted in Section 3. If φ is a formula in $\mathcal{C}^1_{ran}$ over n propositional variables with global clauses $C_1, \ldots, C_l$ where $C_i = \Box(\bigcirc L^i_1 \vee \bigcirc L^i_2 \vee \bigcirc L^i_3)$, then each C_i enforces the constraint $L^i_1 \vee L^i_2 \vee L^i_3 = \text{next}(C_i)$ on the states s_j, $j > 0$, of any interpretation $\mathcal{I}$ which may satisfy φ. With increasing l, the number of distinct valuations which satisfy $\text{next}(C_i)$ for all i, $1 \leq i \leq l$, continuously decreases. For $l/n > 5$, no such satisfying valuation exists for more than 50% of the formulae in $\mathcal{C}^1_{ran}$. This is easy to detect and leads to the good performance of Janssen's algorithm on formulae in this region for $n = 12$ on $\mathcal{C}^1_{ran}$. For

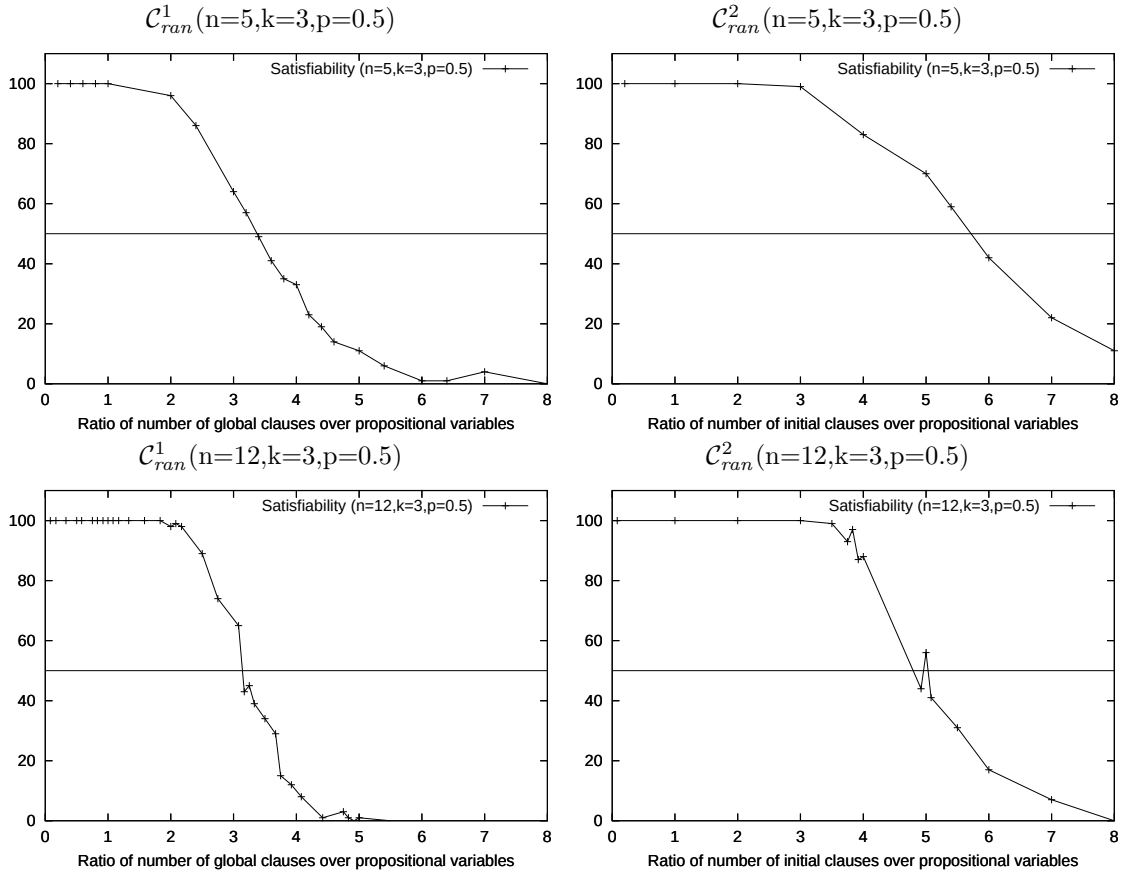

Figure 2: Satisfiability of random PLTL-formulae

all ratios l/n smaller than 5 the sometime clauses in φ have to be taken into account. Consider an arbitrary sometime clause $\Box(\neg p_i \vee \Diamond p_{i+1})$. This clause imposes the constraint $\neg p_i \vee \Diamond p_{i+1}$ on all states in $\mathcal{I}$. To satisfy this disjunction, for s_0, the implementation of Janssen's algorithm in the Logics Workbench 1.1 will always choose to make $\neg p_i$ true, but for all other states this will only be possible if p_i is not already true due to the constraints imposed by the global clauses. Thus, as long as there is a satisfying valuation which makes all next(C_i) true and also all p_i false, $\mathcal{I}$ can be constructed easily. For $n = 12$ we find that for $l/n \leq 0.33$ this is true for more than 50% of the formulae and for $l/n \leq 0.58$ it is still true for more than 40% of the formulae. The percentile graph for $n = 12$ shows quite a good correlation between the sudden rise in the CPU time consumption and the decline of the percentage of formulae for which models can be found in the way described above.

To explain the behaviour of the Logics Workbench for $n = 12$ and ratios $0.83 \leq l/n \leq 5$ we take Theorem 2 into account. The proof of Theorem 2 tells us something about the structure of PLTL-models for random SNF formulae. There are basically two possibilities. Either the model consists of an infinite sequence of states whose identical valuations satisfy the formula ψ_0 defined in the proof of Theorem 2, or the model consists of an infinite repetition of the sequence of states $s_1, \ldots, s_n$ where each s_i satisfies the formulae ψ_i defined above. The first possibility provides an explanation of the good behaviour of Janssen's algorithm for ratios $l/n \leq 0.58$. However, for ratios $0.83 \leq l/n \leq 5$ almost all models will be structured according to the second possibility. Since the the implementation of Janssen's algorithm in the Logics Workbench 1.1. is not aware that the states $s_1, \ldots, s_n$ have to satisfy particular valuations, it searches through up to n^{2^n} combinations of possible valuations, until a model has been found. Since the global clauses put ever stronger constraints on satisfying constraints with increasing ratio l/n, the search space is getting continuously smaller. Thus, the continuous decrease in the median CPU time observed after the peak at $l = 1.17$. Curiously, this peak occurs at a ratio l/n which is not related to the ratio $l/n = 3.15$ where 50% of the formula are satisfiable and 50% of the formulae are unsatisfiable.

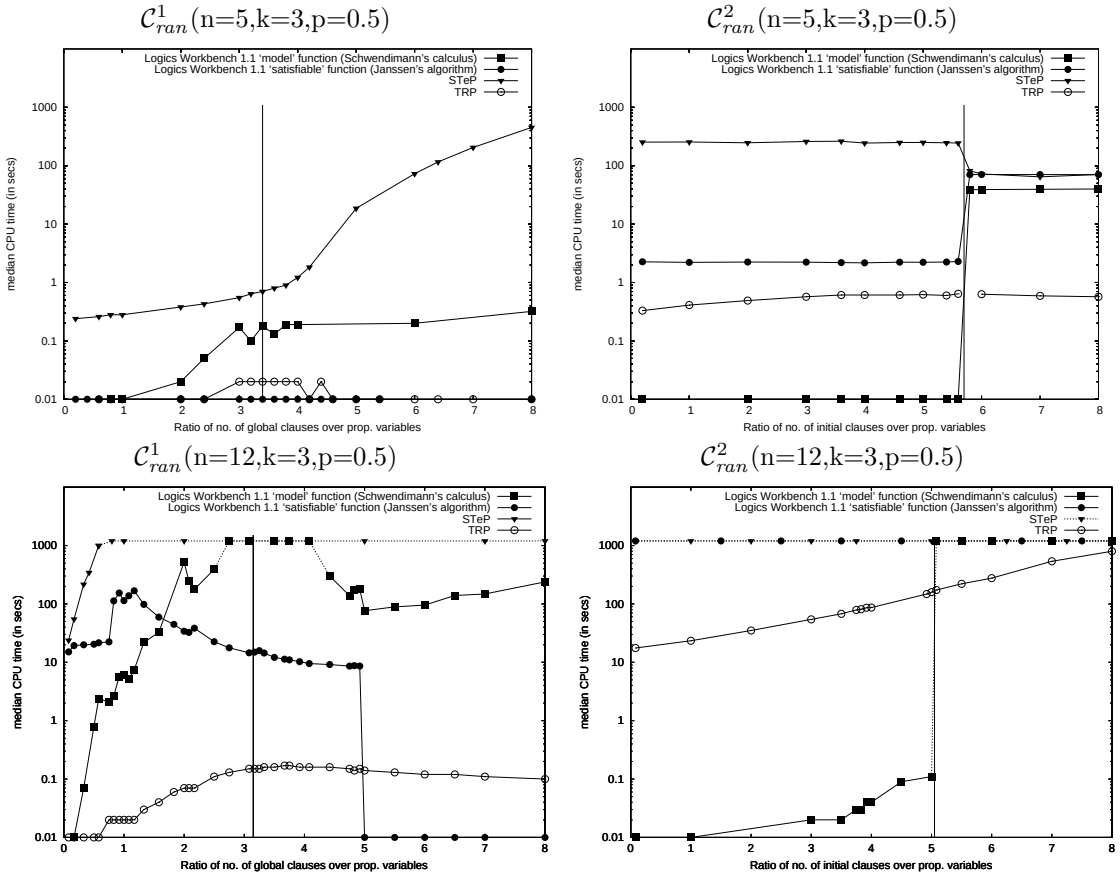

Figure 3: Performance of the decision procedures

As predicted, the performance of *STeP* is worse than that of the Logics Workbench 1.1 'satisfiable' function. We have already observed that on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n < 0.58$ the majority of the formulae under consideration are satisfied by an interpretation consisting of an infinite sequence of states whose identical valuations satisfy the formula ψ_0 defined in the proof of Theorem 2. Only for these formulae *STeP* terminates within the imposed time-limit of 1000 CPU seconds. For all other formulae the simultaneous consideration of all potentially satisfying interpretations in conjunction with a delayed eventuality check provides the explanation for the inferior performance of *STeP*. A surprise, however, is the behaviour of *STeP* on $\mathcal{C}^1_{ran}$ for $n = 5$ and $l/n > 4.5$ where the median CPU time increases. Since the number of potentially satisfying interpretations that need to be considered decreases in this area, we were expecting a much better performance of *STeP*, similar to the good performance we see for the Logics Workbench 1.1 'satisfiable' function for $n = 12$ and $l/n > 5$.

The Logics Worbench 1.1 'model' function behaves uniformly better than *STeP* and is also able to outperform the Logics Workbench 'satisfiable' function on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n < 1.8$. The behaviour of Schwendimann's one-phase tableau calculus is not so much influenced by the total number of models that the formula under consideration has. Instead it is influenced by the number of backtracks that the system needs to perform until a model has been found. This in turn is influenced by the simplification and redundancy elimination rules and by the heuristics which determine the sequence of tableau expansion steps the system performs which all may allow the system to avoid the introduction of superfluous backtrack points. Unfortunately, the implementation of Schwendimann's calculus lacks any simplification and redundancy elimination rules and does not use sophisticated heuristics to guide its search. This becomes evident on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n \geq 5$, where the Logics Worbench 1.1 'model' function should be much faster in determining that the formulae in this region are unsatisfiable.

TRP behaves as predicted. As pointed out above, for $n = 12$ and $l/n > 5$, the conjunction of global clauses in a formula of $\mathcal{C}^1_{ran}$ is unsatisfiable. The slow and steady increase in the time needed to decide $\mathcal{C}^1_{ran}$ for-

mulae for $l/n \geq 5$ is due to the increase in the number of global clauses in the formulae which are saturated under the three inference rules. For $n = 12$ and $l/n < 0.83$ the number of resolvents generated is below 10, reaching about 1000 resolvents for $l/n = 5$, and dropping to below 350 resolvents for $l/n > 6$.

For $\mathcal{C}^2_{ran}$ we see that *TRP* is orders of magnitude slower than the 'model' function of the Logics Workbench 1.1 on all satisfiable formulae generated for $n = 5$ and $n = 12$. This is due to the fact that this algorithm behaves exactly as described in the previous section, given that the clauses of the formulae in $\mathcal{C}^2_{ran}$ are written in a particular order which allows the algorithm to pick the clauses and literals of the formulae in an optimal order. Although this shows that the considerations in the previous section concerning these two decision procedures are correct, it is interesting to see that for $n = 12$ the 'model' function of the Logics Workbench is not able to solve a single unsatisfiable formula in $\mathcal{C}^2_{ran}$. Concerning the Logics Workbench 1.1 'satisfiable' function it is important to point out that for $n = 12$ the algorithm fails to solve a single problem not because it exceeds the time-limit, but because it terminates with a segmentation fault after exhausting the space in one of its internal data structures.

Summing up, for the first class of problems the characteristics of $\mathcal{C}^1_{ran}$ roughly leads to the behaviour predicted in Section 3. For the second class of problems the predicted behaviour is observed only when comparing the curves for *TRP* and the 'model' function implementation of Schwendimann's calculus. The behaviour of the other tableaux implementations is not in accordance with the hypothesis. This indicates that further analysis is necessary before the hypothesis can be regarded as empirically validated. However, the results of the experiments are evidence of a correlation between the structural properties of the problems and the computational behaviour of the implemented decision procedures for PLTL.

References

[1] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "Making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, May 1994.

[2] F. Baader and S. Tobies. The inverse method implements the automata approach for modal satisfiability. In *Proc. IJCAR 2001*, volume 2083 of *LNAI*, pages 92–106. Springer, 2001.

[3] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. AAAI-98*, pages 262–267. AAAI Press, 1998.

[4] H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic Journal of the IGPL*, 8(3):265–292, May 2000.

[5] C. Dixon. Search strategies for resolution in temporal logics. In *Proc. CADE-13*, LNAI 1104, pages 673–687. Springer, 1996.

[6] C. Dixon. Using Otter for temporal resolution. In *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*, pages 149–166. Kluwer, 2000.

[7] F. M. Donini, P. Liberatore, F. Massacci, and M. Schaerf. Generating symbolic model-checking benchmarks by reduction from QBF. In *Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics*, Technical Report DII 14/01, pages 10–18, Italy, 2001. Dipartimento di Ingegneria dell'Informazione, Universitá degli Studi di Siena.

[8] M. Fisher. A resolution method for temporal logic. In *Proc. IJCAI'91*, pages 99–104. Morgan Kaufman, 1991.

[9] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, 2001.

[10] J. Franco and M. Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[11] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proc. AAAI-93*, pages 28–33. AAAI Press, 1993.

[12] A. Goldberg, P. Purdom, and C. Brown. Average time analyses of simplified Davis-Putnam procedures. *Information Processing Letters*, 15(2):72–75, 1982.

[13] A. Goldberg, P. Purdom, and C. Brown. Corrigendum: "Average time analyses of simplified Davis-Putnam procedures", Information Processing Letters 15(2):72–75. *Information Processing Letters*, 16(4):213–213, 1983.

[14] G. D. Gough. Decision procedures for temporal logic. Technical Report UMCS-89-10-1, Department of Computer Science, University of Manchester, UK, 1989.

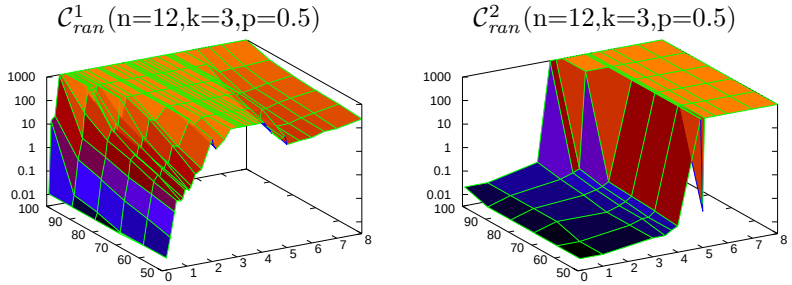

Figure 4: CPU time percentiles for the 'model' function of the Logics Workbench 1.1 (Schwendimann's calculus)

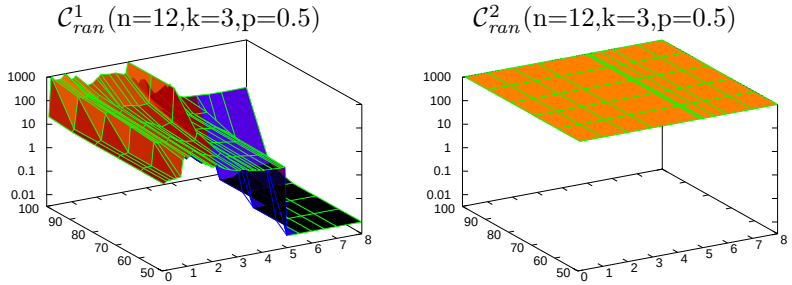

Figure 5: CPU time percentiles for the 'satisfiable' function of the Logics Workbench 1.1 (Janssen's algorithm)

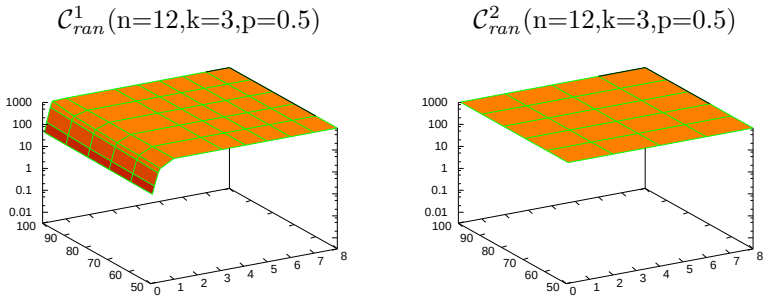

Figure 6: CPU time percentiles for *STeP*

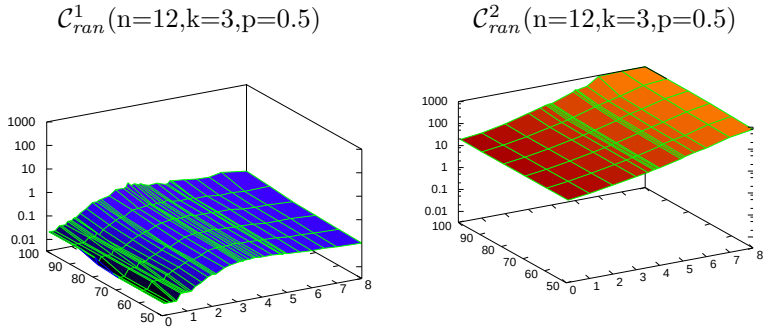

Figure 7: CPU time percentiles for *TRP*

[15] J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42, 1996.

[16] I. Horrocks and P. F. Patel-Schneider. Evaluating optimised decision procedures for propositional modal $\mathbf{K}_{(\mathbf{m})}$ satisfiability. In *SAT2000: Highlights of Satisfiability Research in the Year 2000*. ISO Press, 2000.

[17] I. Horrocks, P. F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[18] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999.

[19] U. Hustadt and R. A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In *Proc. IJCAI'99*, pages 110–115. Morgan Kaufmann, 1999.

[20] U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. In *SAT2000: Highlights of Satisfiability Research in the Year 2000*, pages 459–483. IOS Press, 2000.

[21] G. Janssen. *Logics for Digital Circuit Verification: Theory, Algorithms, and Applications*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1999.

[22] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Proc. CAV'93*, volume 697 of *LNCS*, pages 97–109. Springer, 1993.

[23] Z. Manna and the STeP group. STeP: The Stanford Temporal Prover. Technical report STAN-CS-TR-94-1518, Department of Computer Science, Stanford University, USA, 1994.

[24] P. F. Patel-Schneider and R. Sebastiani. A system and methodology for generating random modal formulae. In *Proc. IJCAR 2001*, volume 2083 of *LNAI*, pages 464–468. Springer, 2001.

[25] J. Rintanen. Improvements to the evaluation of quantified booelan formulae. In *Proc. IJCAI'99*, pages 1192–1197. Morgan Kaufmann, 1999.

[26] S. Schwendimann. *Aspects of Computational Logic*. PhD thesis, Universität Bern, Switzerland, 1998.

[27] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. AAAI-92*, pages 440–446. MIT Press, 1992.

[28] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, 1985.

[29] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.