A *SAT-based* decision procedure for $\mathcal{ALC}$

Fausto Giunchiglia
IRST, 38050 Povo, Trento, Italy.
DISA, Università di Trento, Italy.
fausto@irst.itc.it

Roberto Sebastiani
DIST, Università di Genova,
v. Causa 15, 16146 Genova, Italy.
rseba@mrg.dist.unige.it

Abstract

The goal of this paper is to describe and thoroughly test a decision procedure, called KSAT, checking satisfiability in the terminological logic $\mathcal{ALC}$. KSAT is said to be *SAT-based* as it is defined in terms of a decision procedure for propositional satisfiability (SAT). The tests are performed comparing KSAT with, among other procedures, KRIS, a state-of-the-art tableau-based implementation of a decision procedure for $\mathcal{ALC}$. KSAT outperforms KRIS of orders of magnitude. Furthermore, the empirical results highlight an intrinsic weakeness that tableau-based decision procedures have with respect to SAT-based decision procedures.

1 INTRODUCTION

The goal of this paper is to describe and thoroughly test a new decision procedure, called KSAT, checking satisfiability in the terminological logic $\mathcal{ALC}$, as defined in (Schmidt-Schauß & Smolka 1991), comprising Boolean operations on concepts and value restrictions, and not restricted to CNF form.[1]

As it is well known, $\mathcal{ALC}$ is a notational variant of K(m), that is, K with m modalities (Schild 1991).[2] The main idea underlying the definition of KSAT is that a decision procedure for satisfiability in K(m) (K(m)-satisfiability) can be defined in terms of a decision procedure for propositional satisfiability (SAT). As a matter of terminology, we call *SAT-based* all the decision procedures whose definition is based on this idea.[3]

KSAT outperforms all the decision procedures and systems for terminological and modal logics we have been able to acquire. In this paper we compare KSAT with two of them. The first is a tableau-based procedure — due to B. Nebel and E. Franconi — which is essentially a straightforward implementation of the algorithm described in (Hollunder, Nutt & Schmidt-Schauß 1990). This procedure is called TABLEAU from now on. [4] The second is the state-of-the-art system KRIS described in (Hollunder et al. 1990, Baader, Franconi, Hollunder, Nebel & Profitlich 1994). [5] There are many reasons why a system can be more efficient than another. A crucial one is the "smartness" of the implementation. The implementation of KSAT we use is naive in many respects, e.g., it is in Lisp and it does not use fancy optimized data structures. We still have to push our work in this direction. KSAT is smarter than its competitors for a much more interesting reason. Both TABLEAU and KRIS are tableau-based. As Section 4 describes in detail, tableau-based decision procedures have an intrinsic weakness which makes it very hard if not impossible to be as efficient as SAT-based decision procedures. In our opinion, this is the most interesting theoretical result of this paper.

[1] KSAT, the test code and all the results presented in this paper are available via anonymous FTP at ftp.mrg.dist.unige.it in the directory pub/mrg-systems/ksat/ksat1.

[2] In this paper we always refer to K(m) rather than to $\mathcal{ALC}$. In particular, we speak of wffs rather than concepts, modalities rather than roles, and so on. K(m)'s syntax is simpler than $\mathcal{ALC}$'s. Notice however that the current implementation of KSAT works with $\mathcal{ALC}$'s syntax.

[3] Although this is beyond the goals of this paper, it is worth noticing that this methodology is general, and can be extended to the other normal and (we think) non normal logics, following the methodology and results presented in (Giunchiglia & Serafini 1994, Giunchiglia, Serafini, Giunchiglia & Frixione 1993) (but see also, e.g., (Fitting 1983, Massacci 1994)).

[4] TABLEAU is available via anonymous FTP at ftp.mrg.dist.unige.it in pub/mrg-systems/tableau.

[5] KRIS is available via anonymous FTP at ftp.dfki.uni-sb.de in /pub/tacos/KRIS.

The paper is structured as follows. In Section 2 we present the algorithm implemented by KSAT. In Section 3 we briefly survey our test methodology, originally defined in (Giunchiglia & Sebastiani 1996). This material is needed for a correct understanding of the experimental results reported later. In Section 4 we perform a comparative analysis of a first set of experimental results. This analysis allows us to show why SAT-based decision procedures are intrinsecally more efficient than tableau-based decision procedures. Finally, in Section 5 we perform an exhaustive empirical analysis of KSAT and KRIS, that is, the fastest SAT-based and the fastest tableau-based decision procedure at our disposal. This allows us to confirm the analysis done in Section 4 and, looking at the KSAT results, to get a better understanding of where the hardest cases are. Among other things, this allows us to reveal what looks like a phase transition (Mitchell, Selman & Levesque 1992, Williams & Hogg 1994). To our knowledge this is the first time that this phenomenon has been found in a modal logic.

The analysis presented in this paper builds on and takes to its conclusion the work preliminarily described in (Giunchiglia & Sebastiani 1996). It improves on the previous material in three important aspects. Let us call KSAT_0 the decision procedure presented in (Giunchiglia & Sebastiani 1996) (called KSAT in (Giunchiglia & Sebastiani 1996)).[6] First, the algorithm and its heuristics, are extended from dealing with a single modality to dealing with multiple modalities. Second, the implementation is improved. KSAT is much faster than KSAT_0 (in our tests, up to two orders of magnitude, see Sections 4 and 5). This has been obtained essentially by adding an initial phase of wff preprocessing. Other — relatively minor — implementational variations can be understood by comparing the code of the two systems. Third, and more important, the testing in (Giunchiglia & Sebastiani 1996) was not exhaustive and only compared KSAT_0 with TABLEAU. This made us miss some important points, and the phenomena described in Sections 4 and 5 went unnoticed. Furthermore, the increased efficiency of KSAT_0 with respect to TABLEAU in (Giunchiglia & Sebastiani 1996) was wrongly motivated by the efficiency of the propositional decision procedure. KSAT and KSAT_0 (when applied to a single modality) implement essentially the same algorithm. KSAT is only a more efficient implementation. The same applies to KRIS and TABLEAU. As the results in Section 5 show, the move from KSAT_0 to KSAT, or from TABLEAU to KRIS causes an increase in efficiency, but it does not change the shape of the efficiency curves, as it happens in the move from TABLEAU to KSAT_0 (or from KRIS to KSAT).

[6] KSAT_0, the test code and all the results presented in (Giunchiglia & Sebastiani 1996) are available via anonymous FTP at `ftp.mrg.dist.unige.it` in `pub/mrg-systems/ksat/ksat0`.

2 THE ALGORITHM

Let us write $\Box_r$ to mean the r-th modality. Let us call *atom* any wff which can not be decomposed propositionally, and *modal atom* any atom of the form $\Box_r\psi$. Let φ be the modal wff to be proved satisfiable. The algorithm for testing K(m)-satisfiability follows two basic steps, the first implementing the propositional reasoning, the second implementing the modal reasoning:

1. Using a decision procedure for propositional satisfiability, assign a truth value to (a subset of) the atoms occurring in φ in a way to make φ evaluate to T. Let us call *truth assignment (for φ)* the resulting set μ of truth value assignments. We say that μ *propositionally satisfies* φ.[7] Then μ is of the form

$$
\begin{aligned}
\mu = \{ & \Box_1\alpha_{11} = T, \Box_1\alpha_{12} = T, \ldots, \\
& \Box_1\beta_{11} = F, \Box_1\beta_{12} = F, \ldots, \\
& \ldots \\
& \Box_m\alpha_{m1} = T, \Box_m\alpha_{m2} = T, \ldots, \\
& \Box_m\beta_{m1} = F, \Box_m\beta_{m2} = F, \ldots, \\
& A_1 = T, A_2 = T, \ldots, \\
& A_{R+1} = F, A_{R+2} = F, \ldots \}
\end{aligned}
$$

Notationally, from now on we write μ as

$$
\begin{aligned}
\mu \quad = \quad & \bigwedge_i \Box_1\alpha_{1i} \wedge \bigwedge_j \neg\Box_1\beta_{1j} \wedge \\
& \ldots \\
& \bigwedge_i \Box_m\alpha_{mi} \wedge \bigwedge_j \neg\Box_m\beta_{mj} \wedge \gamma
\end{aligned} \tag{1}
$$

where $\gamma = \bigwedge_{k=1}^{R} A_k \ \wedge \bigwedge_{h=R+1}^{S} \neg A_h$ is a conjunction of propositional literals. Furthermore we use the greek letters μ, η to represent truth assignments.

2. Prove that the input wff φ is K(m)-satisfiable by finding (among all the possible truth assignments) a K(m)-satisfiable truth assignment μ of form as in (1). μ is K(m)-satisfiable iff the restricted assignment

$$
\mu^r = \bigwedge_i \Box_r\alpha_{ri} \wedge \bigwedge_j \neg\Box_r\beta_{rj} \tag{2}
$$

[7] Notice that it is not necessary for a truth assignment to assign *all* the atoms of φ. For instance, $\{\Box_1\psi_1 = T\}$ propositionally satisfies $\Box_1\psi_1 \vee \Box_2\psi_2$.

```
function KSAT(φ)
    return KSAT_W(φ, T);

function KSAT_W(φ, μ)
    if φ = T                                          /* base      */
        then return KSAT_A(μ);
    if φ = F                                          /* backtrack */
        then return False;
    if {a unit clause (l) occurs in φ}                /* unit      */
        then return KSAT_W(assign(l, φ), μ ∧ l);
    l := choose-literal(φ);                           /* split     */
    return  KSAT_W(assign(l, φ), μ ∧ l)  or
            KSAT_W(assign(¬l, φ), μ ∧ ¬l);

function KSAT_A(⋀_i □_1 α_1i ∧ ⋀_j ¬□_1 β_1j ∧ ... ∧ ⋀_i □_m α_mi ∧ ⋀_j ¬□_m β_mj ∧ γ)
    for any box index r do
        if not KSAT_RA(⋀_i □_r α_ri ∧ ⋀_j ¬□_r β_rj)
            then return False;
    return True;

function KSAT_RA(⋀_i □_r α_ri ∧ ⋀_j ¬□_r β_rj)
    for any conjunct "¬□_r β_rj" do
        if not KSAT(⋀_i α_ri ∧ ¬β_rj)
            then return False;
    return True;
```

Figure 1: The basic version of KSAT algorithm.

is K(m)-satisfiable, for every r. μ^r is K(m)-satisfiable iff the wff

$$\varphi^{rj} = \bigwedge_i \alpha_{ri} \wedge \neg\beta_{rj} \qquad (3)$$

is K(m)-satisfiable, for every j. If no truth assignment is found which is K(m)-satisfiable, then φ is not K(m)-satisfiable (K(m)-unsatisfiable).

The two steps recurse until we get to a truth assignment with no modal atoms.

The algorithm is implemented by the function KSAT in Figure 1. KSAT takes in input a modal propositional wff φ and returns a truth value asserting whether φ is K(m)-satisfiable or not. KSAT invokes directly KSAT$_W$(where "$_W$" stands for "Wff"), passing as arguments φ and the truth value T (i.e., by (1), the empty truth assignment). KSAT$_W$ tries to build a K(m)-satisfiable truth assignment μ satisfying φ. This is done recursively, according to the following steps:

- (base) If $\varphi = T$, then μ satisfies φ. Thus, if μ is K(m)-satisfiable, then φ is K(m)-satisfiable. Therefore KSAT$_W$ invokes KSAT$_A(\mu)$ (where "$_A$" stands for (truth) Assignment). KSAT$_A$ returns a truth value asserting whether μ is K(m)-satisfiable or not.
- (backtrack) If $\varphi = F$, then μ can not be a truth assignment for φ. Therefore KSAT$_W$ returns *False*.
- (unit) If a literal l occurs in φ as a unit clause, then l must be assigned T. [8] To obtain this, KSAT$_W$ is invoked recursively with arguments the wff returned by *assign*(l, φ) and the assignment obtained by adding l to μ. *assign*(l, φ) substitutes every occurrence of l in φ with T and evaluates the result.
- (split) If none of the above situations occurs, then *choose-literal*(φ) returns an unassigned literal l according to some heuristic criterion. Then KSAT$_W$ is first invoked recursively with arguments *assign*(l, φ) and $\mu \wedge l$. If the result is negative, then KSAT$_W$ is invoked with arguments *assign*$(\neg l, \varphi)$ and $\mu \wedge \neg l$.

KSAT$_A(\mu)$ invokes KSAT$_{RA}(\mu_r)$ (where "$_{RA}$" stands for Assignment Restricted to one modality) for any index r such that $\Box_r$ occurs in μ. KSAT$_{RA}$ returns a truth value asserting whether μ_r is K(m)-satisfiable or not.

The correctness and completeness of KSAT can be easily seen, for instance by noticing the close parallel with Fitting's tableau described in (Fitting 1983). It is important to notice that KSAT$_W$ is a variant of the

[8] A notion of unit clause for non-CNF propositional wffs is given in (Armando & Giunchiglia 1993). More generally, (Armando & Giunchiglia 1993) and (Sebastiani 1994) show how decision procedures for CNF formulas can be modified to work for non-CNF formulas

non-CNF version of the Davis-Putnam-Longemann-Loveland SAT procedure (Davis & Putnam 1960, Davis, Longemann & Loveland 1962) (DPLL from now on), as described in (Armando & Giunchiglia 1993). Unlike DPLL, whenever an assignment μ has been found, KSAT_W, instead of returning *True*, invokes $\text{KSAT}_A(\mu)$. Essentially, DPLL is used to generate truth assignments, whose K(m)-satisfiability is recursively checked by KSAT_A. We have implemented the algorithm described in Figure 1 as a procedure, also called KSAT, implemented in Common Lisp on top of the non-CNF DPLL decision procedure described in (Armando & Giunchiglia 1993). DPLL is well known to be one of the fastest decision procedures for SAT (see, e.g., (Buro & Buning 1992, Uribe & Stickel 1994)). However the implementation we use, though relatively fast, is much slower than the state-of-the-art SAT decision procedures (see, e.g., (Buro & Buning 1992, Zhang & Stickel 1994)). The basic version of the algorithm described in Figure 1 is improved in the following way. First, all modal atoms are internally ordered. This avoids assigning different truth values to permutations of the same sub-wffs. Secondly, KSAT_{RA} is implemented in such a way to "factorize" the common component $\bigwedge_i \alpha_{ri}$ in searching truth assignments for $\bigwedge_i \alpha_{ri} \wedge \neg\beta_{r1}, \bigwedge_i \alpha_{ri} \wedge \neg\beta_{r2}, \ldots$. Finally, KSAT_W is modified in such a way that KSAT_A is (optionally) invoked on intermediate assignments before every split. This drastically prunes search whenever unconsistent intermediate assignments are detected. These topics are described in detail in (Giunchiglia & Sebastiani 1996). More recently we have also introduced a form of preprocessing — essentially, a recursive removal of duplicate and contradictory subwffs — of the input formulas.

3 THE TEST METHOD

The methodology we use generalizes the *fixed-clause-length* model commonly used in propositional SAT testing (see, e.g., (Mitchell et al. 1992, Buro & Buning 1992)).

Let a $3\text{CNF}_{K(m)}$ wff be a conjunction of $3\text{CNF}_{K(m)}$ clauses. Let a $3\text{CNF}_{K(m)}$ clause be a disjunction of three $3\text{CNF}_{K(m)}$ literals, i.e., $3\text{CNF}_{K(m)}$ atoms or their negations. Let a $3\text{CNF}_{K(m)}$ atom be either a propositional atom or a wff in the form $\Box_r C'$, C' being a $3\text{CNF}_{K(m)}$ clause. Then $3\text{CNF}_{K(m)}$ wffs are randomly generated according to the following parameters:

(i) the modal depth d;
(ii) the number of distinct boxes m;
(iii) the number of clauses L;
(iv) the number of propositional variables N;
(v) the probability p with which any randomly generated $3\text{CNF}_{K(m)}$ atom is propositional. (p estabilishes thus the percentage of propositional atoms at every level of the wff tree.)

Notice that, if we set $d = 0$, we have the standard 3SAT test method (Mitchell et al. 1992).

For fixed N, d, m and p, for increasing values of L, a certain number (100, 500, 1000...) of random $3\text{CNF}_{K(m)}$ wffs are generated, internally sorted, and then given in input to the procedure under test. Satisfiability percentages and mean/median CPU times are plotted against the L/N ratio.

Similarly to the propositional 3CNF case, the methodology proposed above presents three main features. First, the method is very general: $3\text{CNF}_{K(m)}$ wffs represent all K(m) wffs, as there there is a K(m)-satisfiability-preserving way of converting any K(m) wff into $3\text{CNF}_{K(m)}$. Second, the usage of $3\text{CNF}_{K(m)}$ form minimizes the number of parameters to handle. Finally, the parameters L and N allow for a coarse "tuning" of both the satisfiability probability and the hardness of random 3CNF modal wffs, so that it is possible to generate very hard problems with near 0.5 satisfiability probability.

4 TABLEAU-BASED VS. SAT-BASED PROCEDURES

In the tests described in this section we have tested and compared TABLEAU, KRIS, KSAT_0 and KSAT on the same group of 4,000 random formulas, with $d = 2$, $m = 1$, $N = 3$, $p = 0.5$, $L \in \{N \ldots 40N\}$, 100 samples/point. These values have been chosen as in the analysis described in (Giunchiglia & Sebastiani 1996) they gave the highest execution times with both TABLEAU and KSAT_0. The range $N \ldots 40N$ for L has been chosen empirically to cover coarsely the "100% satisfiable – 100% unsatisfiable" transition. As a general test rule we have introduced a timeout of 1000s on each sample wff. If the decision procedure under test exceeds the timeout for a given wff, a failure value is returned and the CPU time value is conventionally set to 1000s. Furthermore, we have stopped running the whole test whenever more than 50% samples (e.g., 50 out of 100 samples) have taken more than 1000s each to execute. These two choices have caused a relevant reduction of the testing time. Figure 2 (left) presents the median CPU time plots for all four systems. (We compare median values rather

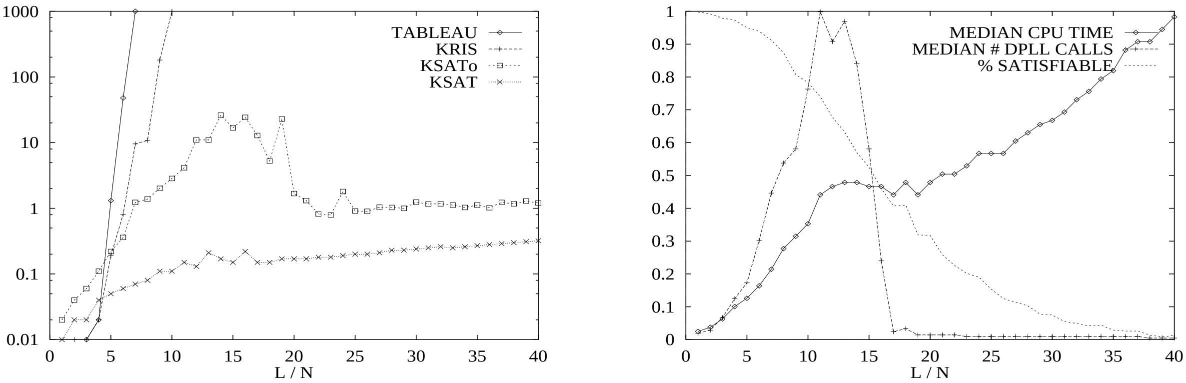

Figure 2: $d = 2$, $m = 1$, $N = 3$, $p = 0.5$, $L = N \ldots 40N$. Left: Tableau, Kris, Ksat$_0$ and Ksat. Median CPU time, 100 samples/point. Right: Ksat. Normalized plots of Median CPU time, Median # of DPLL calls, satisfiability ratio, 1000 samples/point.

than mean values, as the former are much less sensitive to the noise introduced by outliers.) Notice the logarithmic scale on the vertical axis. In Figure 2 (right) we plot respectively the median CPU time, the median number of DPLL calls and the satisfiability percentage curves obtained by running Ksat on the same problem above, with 1000 sample wffs/point. In Figure 2 (right) the curves are all normalized to 1.[9]

Four observations can be made, given below in increasing order of importance.

First, improving the quality of the implementation, e.g., from Tableau to Kris or from Ksat$_0$ to Ksat, may introduce good quantitative performance improvements. In fact, Kris reaches the time bound at the 10th step, while Tableau reaches the time bound at the 7th step, about two orders of magnitude above the corresponding Kris value. Similarly, Ksat$_0$ has a maximum at the 14th step, more than 2 orders of magnitude above the corresponding Ksat value. However, and this is the second observation, improving the quality of the implementation does not seem to affect the qualitative behaviour of the procedures. In fact, both the Tableau and Kris curves present a supposedly exponential growth with the number of clauses, while both Ksat$_0$ and Ksat curves flatten when the number of clauses exceeds a certain value.

Third, independently from the quality of implementation, Ksat and Ksat$_0$ quantitatively outperform Tableau and Kris. For instance, the performance gap between Ksat and Kris at the 10th step is about 4 orders of magnitude. Moreover, the extrapolation of the Kris curve suggests that its value — and the performance gap with Ksat — would reach several orders of magnitude for problems at the right end side of the plots. To support this consideration, we ran Kris on 100 samples of the same problem, for $L = 40N$. No sample wff was solved within the timeout. When releasing the timeout mechanism, Kris was not able to end successfully the computation of the first sample wff after a run of one month. Fourth, and most important, independently of the quality of implementation, Ksat and Ksat$_0$ qualitatively outperform Tableau and Kris. In fact, while Tableau and Kris present a supposedly exponential growth against the number of clauses, both Ksat$_0$ and Ksat curves present a polynomial growth. In particular, the Ksat CPU time curve (like that of Ksat$_0$) results from a combination of (i) a linear component and (ii) an easy-hard-easy component, centered in the satisfiability transition zone. Both components above are straightforward to notice in Figure 2 (right). The former is due to the preprocessing and to the linear-time function *assign*, which is invoked at every DPLL recursive call. The latter represents the number of recursive DPLL calls, i.e., the size of the tree effectively searched.

The quantitative and qualitative performance gaps pointed out by the third and the fourth observation above are very important and deserve some explanation. Let us consider for instance Ksat and Kris. Both procedures work (i) by enumerating truth assignments which propositionally satisfy the input wff φ and (ii) by recursively checking the K(m)-satisfiability of the assignments found. Both algorithms perform the latter step in the same way. The key difference is

[9] The tests in Figures 2 (left) and 5 have been compiled and run under `Allegro CL 4.2` on a `SUN SPARC10 32M` workstation. The test in Figure 2 (right) has been compiled and run under `AKCL 1.623` on another `SUN SPARC10 32M` workstation. The tests in Figure 4 have been compiled and run under `Allegro CL 4.1` on two identical `SUN SPARC2 32M` workstations.

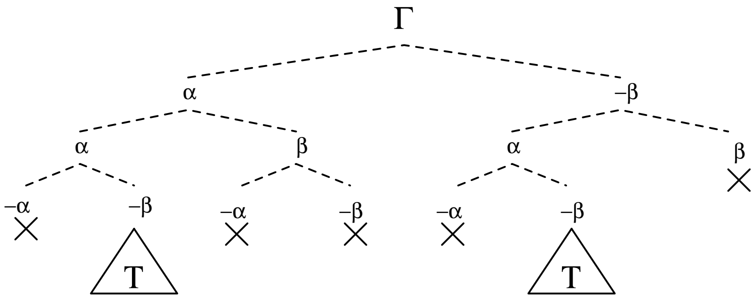

Figure 3: Tableau for the wff $\Gamma = (\alpha \vee \neg\beta) \wedge (\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta)$.

in the first step, that is, in the way KSAT and KRIS handle propositional inference.

In KRIS truth assignments are (implicitly) generated as branches of an analytic propositional tableau, that is, by the recursive application of the rules:

$$\frac{\varphi_1 \wedge \varphi_2}{\varphi_1, \varphi_2}(\wedge\text{-rule}) \quad \frac{\varphi_1 \vee \varphi_2}{\varphi_1 \quad \varphi_2}(\vee\text{-rule}) \qquad (4)$$

and of the other rules for $\{\neg\vee, \neg\wedge, \supset, \neg\supset, \neg\neg\}$. Analytic propositional tableaux perform what we call *syntactic branching*, that is, a branching on the syntactic structure of φ. As widely discussed in (D'Agostino 1992, D'Agostino & Mondadori 1994), any application of the $\vee$-rule generates two subtrees which are *not mutually inconsistent*, [10] that is, two subtrees which may share propositional models. The set of truth assignments enumerated by propositional tableau procedures is intrinsically redundant, and may contain many duplicate and/or subsumed assignments. As a consequence, the number of truth assignments generated grows exponentially with the number of disjunctions occurring positively in φ (in our tests, the number of clauses L), although the actual number of non-redundant assignments propositionally satisfying φ is much smaller. This redundancy is a source of a high degree of inefficiency when using analytic tableaux for propositional satisfiability.

Things get much worse in the modal case. Unlike the propositional case — where tableaux look for *one* assignment satisfying the input formula — in K(m) propositional tableaux enumerate all the truth assignments, which must be recursively checked for K(m)-consistency. (The number of assignments may be huge: up to ten thousands in our tests.) This requires checking recursively (possibly many) subwffs of the form $\bigwedge_i \alpha_{ri} \wedge \beta_j$ of depth $d-1$, for which a propositional tableau will enumerate truth assignments, and so forth. Any redundant truth assignment enumerated at depth d introduces a redundant modal search tree of depth d. Even worse, this propositional redundancy propagates exponentially with the depth d, following the analysis of the subwffs of decreasing depth.

Example 4.1 Consider the simple wff

$$\Gamma = (\alpha \vee \neg\beta) \wedge (\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta),$$

where α and β are modal atoms, and let d be the depth of Γ. The only possible assignment satisfying Γ is $\mu = \alpha \wedge \neg\beta$. Look at Figure 3. The $\vee$-rule is applied to the three clauses occurring in Γ in the order they are listed, and two distinct but identical open branches are generated, both representing the assignment μ. Suppose now that μ is not K(m)-consistent. Then the tableau expands the two open branches in the same way, until it generates two identical (and possibly big) closed modal sub-tableaux T of depth d, each proving the K(m)-inconsistency of μ. This phenomenon may repeat itself at the lower level in each sub-tableaux T, and so forth. For instance, if $\alpha = \Box((\alpha' \vee \neg\beta') \wedge (\alpha' \vee \beta'))$ and $\beta = \Box(\alpha' \wedge \beta')$, then at the lower level we have a wff Γ' of depth $d-1$ analogous to Γ. This propagates exponentially the redundancy with the depth d.

Notice that, if we considered the wff

$$\Gamma^K = \bigwedge_{i=1}^{K} (\alpha_i \vee \neg\beta_i) \wedge (\alpha_i \vee \beta_i) \wedge (\neg\alpha_i \vee \neg\beta_i),$$

the tableau would generate 2^K identical truth assignments $\mu^K = \bigwedge_i \alpha_i \wedge \neg\beta_i$, and things would get exponentially worse. $\Box$

In SAT-based procedures truth assignments are gen-

[10] As pointed out in (D'Agostino 1992, D'Agostino & Mondadori 1994), in Analytic tableaux rules are unable to represent *bivalence*: "every proposition is either true or false, *tertium non datur*". This is a consequence of the elimination of the cut rule in cut-free sequent calculi, from which analytic tableaux are derived.

erated one-shot by a SAT decision procedure. [11] SAT-based procedures perform a search based on what we call *semantic branching*, that is, a branching on the truth value of proper subwffs of φ. Every branching step generates two *mutually unconsistent* subtrees. Because of this, SAT procedures always generate non-redundant sets of assignments. This avoids any search duplication and, recursively on d, any exponential propagation of inefficiency.

Example 4.2 Consider the wff Γ in Example 4.1. A SAT-based procedure branches asserting $\alpha = T$ or $\alpha = F$. The first branch generates $\alpha \wedge \neg\beta$, while the second gives $\neg\alpha \wedge \neg\beta \wedge \beta$, which immediately closes. Therefore, only one instance of the assignment $\mu = \alpha \wedge \neg\beta$ is generated. The same applies recursively to μ^K. □

A propositional wff φ can be seen in terms of a set of constraints for the truth assignments which possibly satisfy it (see, e.g., (Williams & Hogg 1994)). For instance, a clause $A_1 \vee A_2$ constrains every assignment not to set both A_1 and A_2 to F. Unlike tableaux, in SAT procedures branches are cut as soon as they violate some constraint of the wff. The more constrained the wff is, the more likely a truth assignment violates some constraint. (For instance, the bigger is L in a CNF wff, the more likely an assignment generates an empty clause.) Therefore, as φ becomes highly constrained (e.g., when L is big enough) the search tree is very heavily pruned. As a consequence, for L bigger than a certain value, the size of the search tree *decreases* with L, as it can be easily noticed in Figure 2 (right).

5 AN EXAUSTIVE EMPIRICAL ANALYSIS

In the tests described in this section we have tested and compared KSAT and KRIS, that is, the fastest SAT-based and the fastest tableau-based decision procedure at our disposal. We have performed three experiments on 48,000 randomly generated wffs, run according to our test methodology, whose results are all described in Figure 4. All curves represent 100 samples/point. As above, the range $N \ldots 40N$ for L has been chosen empirically to cover coarsely the "100% satisfiable – 100% unsatisfiable" transition. In each experiment we investigate the effects of varying one parameter while fixing the others. In Experiment 1 (left column) we fix $d = 2$, $m = 1$, $p = 0.5$ and plot different curves for increasing numbers of variables $N = 3, 4, 5$. [12] In Experiment 2 (center column) we fix $d = 2$, $N = 4$, $p = 0.5$ and plot different curves for increasing number of distinct modalities $m = 1, 2, 5, 10, 20$. In Experiment 3 (right column) we fix $m = 1$, $N = 3$, $p = 0.5$ and plot different curves for increasing modal depths $d = 2, 3, 4, 5$. For each experiment, we present three distinct sets of curves, each corresponding to a distinct row. In the first (top row) we plot the median CPU time obtained by running both KSAT and KRIS. This gives an overall picture of the qualitative behaviour of KSAT and KRIS and allows for a direct comparison between them. In the second (middle row) we plot the KSAT median number of recursive DPLL calls, that is, the size of the space effectively searched. This allows us to drop the linear component due to the preprocessing and the function calls to *assign*. In the third (bottom row) we plot the percentage of satisfiable wffs evaluated by KSAT. This gives a coarse indication of the average level of constraintness of the test wffs.[13]

Despite the big noise, due to the small samples/point rate (100), the results indicated in Figure 4 provide interesting indications. We report below (Subsection 5.1) a first pass, experiment by experiment, analysis of the results. This gives us an idea of how efficiency and satisfiability are affected by each single parameter. In Subsection 5.2 we report a global and, in some respects, more interesting analysis of the results we have.

5.1 A TESTWISE ANALYSIS

The results of the first experiment (left column) show that increasing N (and L accordingly) causes a relevant increase in complexity — up to one order of magnitude per variable in the "hard" zone for KSAT, up to two orders of magnitude per variable, as far as we can see, for KRIS. This should not be a surprise, as in K/K(m), adding few variables may cause an ex-

[11] In KSAT we used non-CNF DPLL, but we could use any other SAT procedures not affected by the problem highlighted in (D'Agostino 1992, D'Agostino & Mondadori 1994), e.g., OBDDs (Bryant 1992), or an implementation of KE (D'Agostino & Mondadori 1994).

[12] If we compare the KSAT and KRIS plots in Figure 2 (left) with the $L = 3$ KSAT and KRIS plots in Figure 4 (top left), we notice that the plots are different, although they are computed on sample wffs with the same parameter values. This is due to the fact that (i) the former ones are run on a much faster machine; (ii) the starting seeds are different, causing thus the generation of different sample sets.

[13] This percentage is evaluated on the number of samples which *effectively* ended computation within the timeout. Therefore this datum should be considered only as a coarse indication. To obtain an accurate evaluation, we should drop the timeout mechanism and evaluate the satisfiability percentage on at least 1000 samples/point, like in Figure 2 (right).

1: varying variable #: $d = 2, m = 1, p = 0.5$, $N = 3, 4, 5$.

2: varying modality #: $d = 2, N = 4, p = 0.5$, $m = 1, 2, 5, 10, 20$.

3: varying modal depth: $m = 1, N = 3, p = 0.5$, $d = 2, 3, 4, 5$.

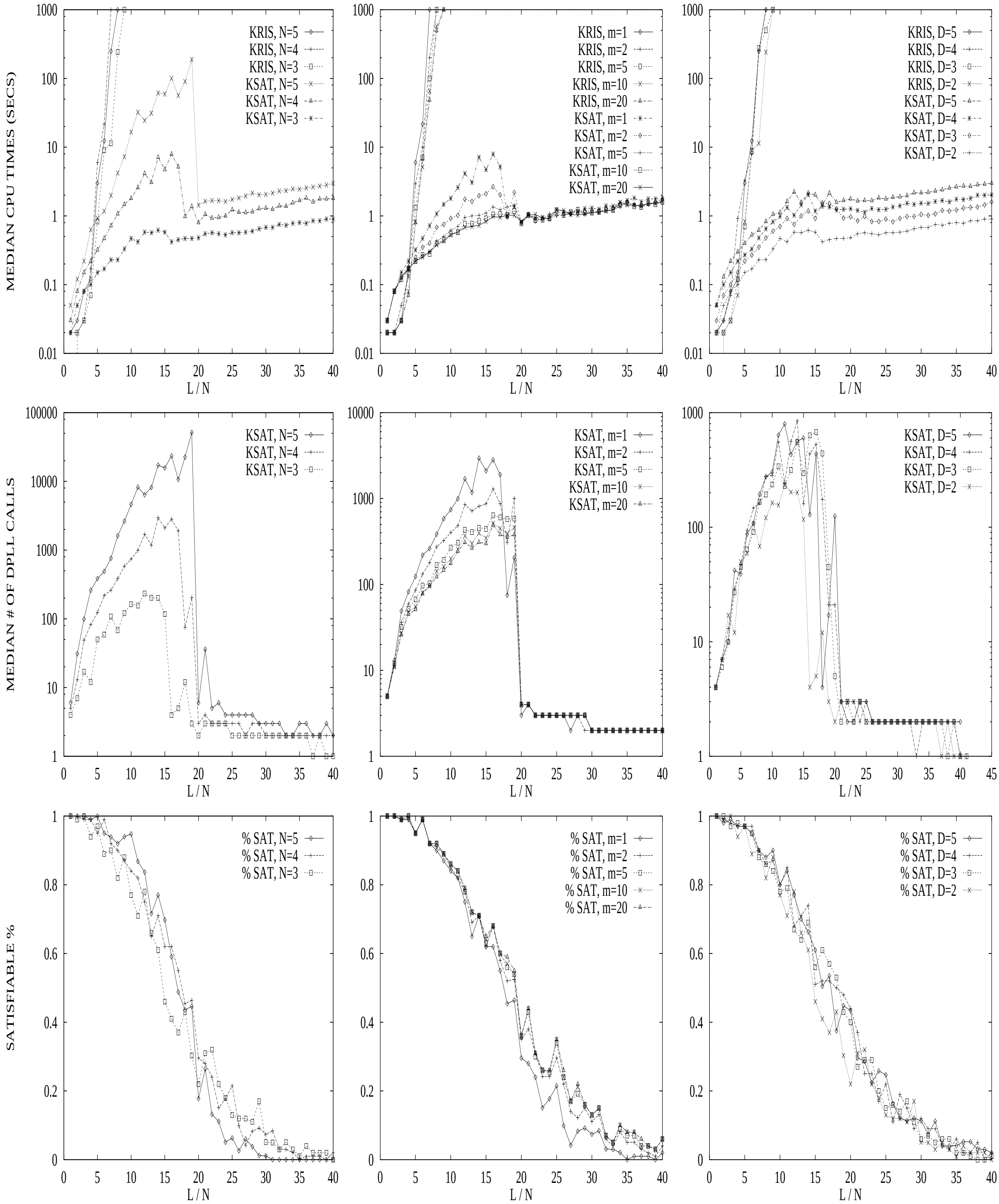

Figure 4: The results of the three experiments.

ponential increase of the search space. Each variable may in fact assume distinct truth values inside distinct states/possible worlds, that is, each variable must be considered with an "implicit multiplicity" equal to the number of states of a potential Kripke model.

The results of the second experiment (center column) present two interesting aspects. First, the complexity of the search monotonically decreases with the increase of the number m of modalities, for both KSAT and KRIS (top and middle box). At a first sight it may sound like a surprise, but it should not be so. In fact, each truth assignment μ is partitioned into m independent sub-assignments μ_r's, each restricted to a single $\Box_r$ (see Equations (1) and (2)). This means "dividing and conquering" the search tree into m non-interfering search trees. Therefore, the bigger is m, the more partitioned is the search space, and the easier is the problem to solve. Second, a careful look reveals that the satisfiability percentage increases with m. Again, there is no mutual dependency between the satisfiability of the distinct μ_r's. Therefore the bigger is m, the less constrained is μ, and the more likely satisfiable is φ.

The results of the third experiment (right column) provide evidence of the fact that the complexity increases with the modal depth d, for both KSAT and KRIS. This is rather intuitive: the higher is d, the deeper are the Kripke models to be searched, and the higher is the complexity of the search.

5.2 A GLOBAL ANALYSIS

The KSAT curves (top and middle rows) highlight the existence a linear and an easy-hard-easy component. In fact, if we increase N from 3 to 5 and L accordingly (left column), the size of the search space has a relevant increase. Therefore, while for $N = 3$ the linear component prevails, for $N = 5$ the easy-hard-easy component dominates. Moreover, when varying the number of modalities (center column), the wff sizes are kept the same for all curves. Therefore, when the effect of the easy-hard-easy component vanishes ($L/N > 20$), the curves collapse together, as the time for preprocessing and *assign* does not depend on the number of modalities m. Notice that the locations of the easy-hard-easy zones do not seem to vary significantly, neither with the number of variables N (left column), nor with the number of modalities m (center column), nor with the depth d (right column).

Let us now consider the satisfiability plots in Figure 4 (bottom row). Despite the noise and the approximations due to timeouts, it is easy to notice that the 50% satisfiability point is centered around $L = 15N \sim 20N$

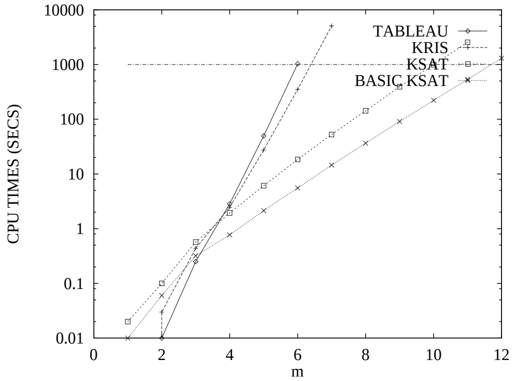

Figure 5: CPU times for the class of φ_d^K formulas.

in all the experiments. Moreover, in the first experiment a careful look reveals that the satisfiability transition becomes steeper when increasing N (e.g., compare the $N = 3$ and $N = 5$ plots). Finally, in all experiments, the curves representing the median number of DPLL calls (middle row) generally locate the peaks around the satisfiability transition, although they seem to anticipate a little the 50% crossover point. From these facts we may conjecture (to be verified!) the existence for K(m)/$\mathcal{ALC}$ of a phase transition phenomenon, similar to that already known for SAT and other NP-hard problems (see, e.g, (Cheeseman, Kanefski & Taylor 1991, Mitchell et al. 1992, Kirkpatrick & Selman 1994)).

The final observation comes from the three sets of median CPU times curves (top row): KSAT outperforms KRIS in all the testbeds, independently on the number of variables N, the number of modalities m or the depth d considered. This confirms the analysis done in Section 4. Again, this is not only a quantitative performance gap (up to 3-4 orders of magnitude) but also a qualitative one, as all KRIS curves grow (supposedly) exponentially with L, while all KSAT curves grow polinomially. To provide further evidence, we have performed another, quite different, test, based on the class of wffs $\{\varphi_d^K\}_{d=1,2,\ldots}$ presented in (Halpern & Moses 1992). This is a class of K(1)-satisfiable wffs, with depth d and $2d + 1$ propositional variables. These wffs are paradigmatic for modal K, as every Kripke structure satisfying φ_d^K has at least $2^{d+1} - 1$ distinct states, while $|\varphi_d^K|$ is $O(d^2)$. From the results in (Halpern & Moses 1992) we can reasonably assume a minimum exponential growth factor of 2^d for any ordinary algorithm based on Kripke semantics. We run TABLEAU, KRIS, KSAT and a "basic" version of KSAT (i.e., with no factorization of $\bigwedge_i \alpha_{ri}$'s and no checking of intermediate assignments), called below BASIC KSAT, on these formulas, for increasing values of d.

The results are plotted in Figure 5. The TABLEAU, KRIS, KSAT and BASIC KSAT curves grow exponentially, approximatively as $(16.0)^d$, $(12.7)^d$, $(2.6)^d$ and $(2.4)^d$ respectively, exceeding 1000s for $d = 6$, $d = 7$, $d = 11$ and $d = 12$ respectively. The slight difference between KSAT and BASIC KSAT is due to the overhead introduced by the $\bigwedge_i \alpha_{ri}$ factorization, which is useless with these formulas. It is worth observing that the result of tracing the global number of truth assignments μ, recursively found by both KSAT and BASIC KSAT, gave exactly $2^{d+1} - 1$ for every d, that is the minimum number of Kripke states. KSAT and BASIC KSAT found no redundant truth assignments.

6 CONCLUSIONS

This paper presents what we think are three very important results:

1. it provides a new implemented algorithm, KSAT, for deciding satisfiability in $\mathcal{ALC}$ (K(m)) which outperforms of orders of magnitude the previous state-of-the-art decision procedures;
2. it shows that the results provided are not by chance, and that all SAT-based modal decision procedures (that is, all the modal decision procedures based on SAT decision procedures) are intrinsically bound to be more efficient than tableau-based decision procedures; and
3. it provides evidence, though very partial, of an easy-hard-easy pattern independent of all the parameters of evaluation considered. If the current partial evidence is confirmed, this is the first time that this phenomenon, well known for SAT and other NP-hard problems, is found in modal logics.

Acknowledgements

Franz Baader, Marco Cadoli, Enrico Franconi, Enrico Giunchiglia, Fabio Massacci and Bernhard Nebel have given very useful feedback. Fabio Massacci has suggested testing the Halpern & Moses formulas. Marco Roveri has given technical assistance in the testing phase. All the members of the Mechanized Reasoning Group in Genoa have put up with many weeks of CPU-time background processes.

References

Armando, A. & Giunchiglia, E. (1993), 'Embedding Complex Decision Procedures inside an Interactive Theorem Prover', *Annals of Mathematics and Artificial Intelligence* **8**(3–4), 475–502.

Baader, F., Franconi, E., Hollunder, B., Nebel, B. & Profitlich, H. (1994), 'An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on', *Applied Artificial Intelligence. Special Issue on Knowledge Base Management* **4**, 109–132.

Bryant, R. E. (1992), 'Symbolic Boolean manipolation with ordered binary-decision diagrams', *ACM Computing Surveys* **24**(3), 293–318.

Buro, M. & Buning, H. (1992), Report on a SAT competition, Technical Report 110, University of Paderborn, Germany.

Cheeseman, P., Kanefski, B. & Taylor, W. M. (1991), Where the really hard problem are, *in* 'Proc. of the 12th International Joint Conference on Artificial Intelligence', pp. 163–169.

D'Agostino, M. (1992), 'Are Tableaux an Improvement on Truth-Tables?', *Journal of Logic, Language and Information* **1**, 235–252.

D'Agostino, M. & Mondadori, M. (1994), 'The Taming of the Cut.', *Journal of Logic and Computation* **4**(3), 285–319.

Davis, M., Longemann, G. & Loveland, D. (1962), 'A machine program for theorem proving', *Journal of the ACM* **5**(7).

Davis, M. & Putnam, H. (1960), 'A computing procedure for quantification theory', *Journal of the ACM* **7**, 201–215.

Fitting, M. (1983), *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Publishg.

Giunchiglia, F. & Sebastiani, R. (1996), Building decision procedures for modal logics from propositional decision procedures - the case study of modal K, *in* 'Proc. of the 13th Conference on Automated Deduction', Lecture Notes in Artificial Intelligence, Springer–Verlag.

Giunchiglia, F. & Serafini, L. (1994), 'Multilanguage hierarchical logics (or: how we can do without modal logics)', *Artificial Intelligence* **65**, 29–70. Also IRST-Technical Report 9110-07, IRST, Trento, Italy.

Giunchiglia, F., Serafini, L., Giunchiglia, E. & Frixione, M. (1993), Non-Omniscient Belief as Context-Based Reasoning, *in* 'Proc. of the 13th International Joint Conference on Artificial Intelligence', Chambery, France, pp. 548–554. Also IRST-Technical Report 9206-03, IRST, Trento, Italy.

Halpern, J. & Moses, Y. (1992), 'A guide to the completeness and complexity for modal logics of knowledge and belief', *Artificial Intelligence* **54**(3), 319–379.

Hollunder, B., Nutt, W. & Schmidt-Schauß, M. (1990), Subsumption Algorithms for Concept Description Languages, *in* 'Proc. 8th European Conference on Artificial Intelligence', pp. 348–353.

Kirkpatrick, S. & Selman, B. (1994), 'Critical behaviour in the satisfiability of random boolean expressions', *Science* **264**, 1297–1301.

Massacci, F. (1994), Strongly analytic tableaux for normal modal logics, *in* 'Proc. of the 12th Conference on Automated Deduction'.

Mitchell, D., Selman, B. & Levesque, H. (1992), Hard and Easy Distributions of SAT Problems, *in* 'Proc. of the 10th National Conference on Artificial Intelligence', pp. 459–465.

Schild, K. D. (1991), A correspondence theory for terminological logics: preliminary report, *in* 'Proc. of the 12th International Joint Conference on Artificial Intelligence', Sydney, Australia, pp. 466–471.

Schmidt-Schauß, M. & Smolka, G. (1991), 'Attributive Concept Descriptions with Complements', *Artificial Intelligence* **48**, 1–26.

Sebastiani, R. (1994), 'Applying GSAT to Non-Clausal Formulas', *Journal of Artificial Intelligence Research* **1**, 309–314. Also DIST-Technical Report 94-0018, DIST, University of Genova, Italy.

Uribe, T. E. & Stickel, M. E. (1994), Ordered Binary Decision Diagrams and the Davis-Putnam Procedure, *in* 'Proc. of the 1st International Conference on Constraints in Computational Logics'.

Williams, C. P. & Hogg, T. (1994), 'Exploiting the deep structure of constraint problems', *Artificial Intelligence* **70**, 73–117.

Zhang, H. & Stickel, M. (1994), Implementing the Davis-Putnam algorithm by tries, Technical report, University of Iowa.