A new method for testing decision procedures in modal logics *

Fausto Giunchiglia[1,2] Marco Roveri[3] Roberto Sebastiani[3]

[1] IRST, 38050 Povo (TN), Italy. ph: ++39.461.314517
[2] DISA, via Inama 5, 38100 Trento, Italy.
[3] DIST, v. Causa 13, 16146 Genoa, Italy. ph: ++39.10.3532811

fausto@irst.itc.it {marco,rseba}@mrg.dist.unige.it

1 Introduction

In the area of automated reasoning in modal logics there seems to be very little bibliography on both problem sets and test methodologies. Most authors do not present empirical tests. Some authors test their systems on groups of single formulas, mostly taken from textbooks (see, e.g., [3, 4]). A few others use formulas which are derived from or simulate realistic problems (see, e.g., [1]). As far as we know, there has been little study on classes of test formulas and on their intrinsic properties (e.g., hardness, satisfiability likelihood). Modal S4 is a noteworthy exception, as it is possible to exploit the Goedel-Tarski translation from intuitionistic logic into S4. A second noteworthy exception can be found in [8], which provides a class of modal formulas which are intrinsically hard, as they are provably satisfiable only in exponential-size Kripke structures.

In this paper we describe a new procedure and, more generally, a new methodology, for testing decision procedures for modal logics, in particular for K and K(m). This procedure is a generalization of the *fixed-clause-length* test methodology, commonly used in propositional satisfiability (see, e.g., [10, 2]). It is easy to implement and to use, and it allows for a statistical control of some important features of the formulas generated, e.g, the hardness and satisfiability rate. [1] In [5, 6] comparative tests on the KSAT and KRIS procedures, performed according to the methodology described in this paper, allowed us to detect two intrinsic weaknesses of tableau-based procedures.

2 The $3\mathrm{CNF}_{K(m)}$ test method

Consider the modal logic K(m), as it is described, e.g., in [8]. A $\mathrm{CNF}_{K(m)}$ wff is a conjunction of $\mathrm{CNF}_{K(m)}$ clauses; a $\mathrm{CNF}_{K(m)}$ clause is a disjunction of

* This paper is a short and slighty revised version of the paper "A new method for testing decision procedures in modal and terminological logics" published as IRST-Technical Report 9609-05. This paper was also presented in the 1996 International Workshop on Description Logics - DL'96.

[1] The code of the test procedure, together with all the empirical results presented in [5, 6], are available via anonymous FTP at `ftp.mrg.dist.unige.it` in `pub/mrg-systems/ksat/`. The papers [5, 6] are available at the same site in `pub/mrg-ftp/`.

```
function rand_wff(d,m,L,N,p)
    for i := 1 to L do
        C_i := rand_cl(d,m,N,p);
    return ⋀_{i=1}^{L} C_i;

function rand_cl(d,m,N,p)
    for j := 1 to 3 do
        l_j := rand_lit(d,m,N,p);
    return ⋁_{j=1}^{3} l_j;

function rand_lit(d,m,N,p)
    φ := rand_atom(d,m,N,p);
    if flip_coin(0.5)
    then return φ;
    else  return ¬φ;

function rand_atom(d,m,N,p)
    if (d=0 or flip_coin(p))
    then return rand_prop_atom(N);
    else  □_r := rand_box(m);
          C := rand_cl(d-1,m,N,p);
          return □_r C ;
```

Fig. 1. The algorithm of the random generator.

$\mathrm{CNF}_{K(m)}$ literals, i.e., $\mathrm{CNF}_{K(m)}$ atoms or their negations; a $\mathrm{CNF}_{K(m)}$ atom is either a propositional atom or a wff of the form $\Box_r C$, where $r \in \{1, \ldots, m\}$ and C is a $\mathrm{CNF}_{K(m)}$ clause. Notice that conjunctions appear only at the top level of a $\mathrm{CNF}_{K(m)}$ wff. A $\mathrm{CNF}_{K(m)}$ wff is called a $3\mathrm{CNF}_{K(m)}$ wff if and only if the number of literals per clause is fixed to 3. $3\mathrm{CNF}_{K(m)}$ wffs can be randomly generated according to 5 parameters: (i) the modal depth d; (ii) the number of distinct boxes m; (iii) the number of top-level clauses L; (iv) the number of propositional variables N; (v) the probability p that a $3\mathrm{CNF}_{K(m)}$ atom is propositional.

The algorithm for randomly generating $3\mathrm{CNF}_{K(m)}$ wffs is presented in Figure 1. A $3\mathrm{CNF}_{K(m)}$ modal wff of depth d is produced by randomly generating L modal $3\mathrm{CNF}_{K(m)}$ clauses of depth d. A $3\mathrm{CNF}_{K(m)}$ modal clause of depth d is produced by randomly generating three $3\mathrm{CNF}_{K(m)}$ modal atoms of depth d, and negating each of them with probability 0.5. (The function *flip_coin(p)* returns $True$ with probability p, $False$ otherwise.) A $3\mathrm{CNF}_{K(m)}$ modal atom of depth d is produced in the following way. If either $d = 0$ or *flip_coin(p)* returns $True$, then the function *rand_prop_atom(N)* picks randomly an atom $A_k \in \{A_1, \ldots, A_N\}$, which is returned. (Intuitively, the parameter p establishes the mean ratio of the propositional atoms at every level of the wff tree.) Otherwise a box $\Box_r$, picked randomly from $\{\Box_1, \ldots, \Box_m\}$ by the function *rand_box(m)*, followed by a $3\mathrm{CNF}_{K(m)}$ modal clause of depth $d-1$, is returned.

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

This method can be extended in a natural way to logics which extend the syntax of K(m) with other prefixed constructs $\mathcal{C}_i$'s, e.g., $*$-iteration, sequence, test .... For every new construct $\mathcal{C}_i$ a function *rand_*$\mathcal{C}_i$*(*$v_1^{\mathcal{C}_i}, v_2^{\mathcal{C}_i}, \ldots$*)* is needed, which we use to randomly generate $\mathcal{C}_i$ according to some specific parameters $v_1^{\mathcal{C}_i}, v_2^{\mathcal{C}_i}, \ldots$. The random generator *rand_wff* is then extended by simply substituting *rand_box(m)* with a function which either (i) selects, with probability $P_{\mathcal{C}_i}$, one $\mathcal{C}_i$, and returns *rand_*$\mathcal{C}_i$*(*$v_1^{\mathcal{C}_i}, v_2^{\mathcal{C}_i}, \ldots$*)*, or (ii) returns *rand_box(m)*, with prob-

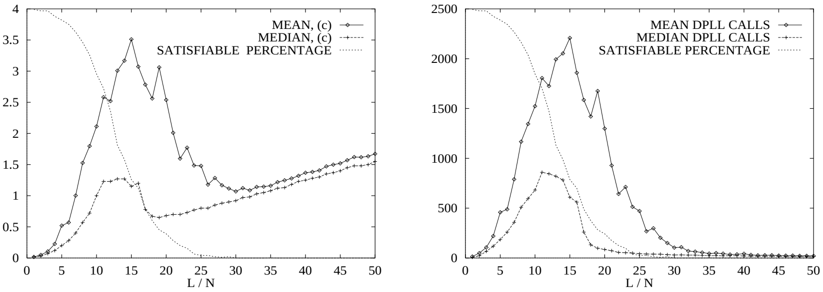

Fig. 2. Testing the KSAT decision procedure [5] by the $3\mathrm{CNF}_{K(m)}$ method, with $m = 1$, $d = 1$, $p = 0.5$, $N = 4$, $L/N = 1 \ldots 50$, 1000 samples/point. Left: mean and median CPU times. Right: mean and median size of search space. Background, left and right: satisfiability percentages (100% at end-scale).

ability $1 - \sum_i P_{\mathcal{C}_i}$. The parameters $P_{\mathcal{C}_i}$, $v_1^{\mathcal{C}_i}, v_2^{\mathcal{C}_i}, \ldots$ are added to the parameter list of the generator, for every construct $\mathcal{C}_i$. To perform a test, all parameters except L and one parameter $v \in \{d, m, p, N, P_{\mathcal{C}_1}, v_1^{\mathcal{C}_1}, v_2^{\mathcal{C}_1}, \ldots\}$ are set to fixed values. Then the experiment can be run similarly to that for K(m).

3 Features of the test method

In [6] we give empirical evidence about how the characteristics of random $3\mathrm{CNF}_{K(m)}$ wffs are affected by the parameters L, N, m and d. First, increasing N (and L accordingly) causes a big increase of the size of the search space. In fact, each variable must be considered with an "implicit multiplicity" equal to the number of states of a potential Kripke model for the input formula φ, which is $O(|\varphi|^d)$ [7]. Second, the complexity of the search monotonically decreases with the number m of modalities. In fact, the search tree is "divided and conquered" into m non-interfering search trees, each restricted to a single $\Box_r$. Therefore, the bigger is m, the more partitioned is the search space, the easier is the solution of the problem. Moreover, the satisfiability percentage increases with m, as the bigger is m, the less constrained — and thus the more likely satisfiable — are the randomly-generated wffs. Third, the size of the search space increases with the modal depth d. In fact, the higher is d, the deeper are the Kripke models to be searched, and the higher is the complexity of the search.

On the whole, the test methodology proposed presents three main features. First, $3\mathrm{CNF}_{K(m)}$ wffs represent all K(m) wffs, as there is a K(m)-satisfiability-preserving way of converting any K(m) wff into $3\mathrm{CNF}_{K(m)}$ [5]. Second, the usage of the $3\mathrm{CNF}_{K(m)}$ form minimizes the number of parameters to handle. In fact, only one "and-branch" parameter L and no "or-branch" parameters are needed, independent of the modal depth d. Finally, the parameters L and N allow for a coarse "tuning" of both the satisfiability probability and the hardness of random $3\mathrm{CNF}_{K(m)}$ wffs. The latter feature is very important and deserves further explanation. Look, e.g., at Figure 2 (taken from [5]). The parameter L monotonically increases the level of "constraintness" of random $3\mathrm{CNF}_{K(m)}$ wffs. Thus, varying the L/N ratio, we pass from a situation where wffs are under-constrained (and thus mostly satisfiable) to one where wffs are over-constrained (and thus mostly unsatisfiable). As a consequence, the plot of the satisfiability percentages draws a transition from 100% satisfiability to 100% unsatisfiability. Moreover, the mean and median size of the search space plots (see Figure 2 right) reveal an easy-hard-easy pattern always centered in the "100% satisfiable-100% unsatisfiable" transition zone. The mean and median CPU time plots reveal also a linear component due to the proper characteristics of KSAT procedure [5, 6] (see Figure 2 left). With increasing N the plots become sharper [6]. This phenomenon has been widely investigated both empirically (see, e.g., [9]) and theoretically (see, e.g., [11]) for many other NP-hard problems. In [5, 6] we revealed it for modal logics by using the $3\text{CNF}_{K(m)}$ method. As a practical consequence, for any value of m, d and p, by suitable choices of L and N, it is possible to generate very hard $3\text{CNF}_{K(m)}$ wffs with near 50% satisfiability probability.

References

1. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. 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, 1994.
2. M. Buro and H. Buning. Report on a SAT competition. Technical Report 110, University of Paderborn, Germany, November 1992.
3. L. Catach. TABLEAUX: a general theorem prover for modal logics. *Journal of Automated Reasoning*, 7, 1991.
4. S. Demri. Uniform and Nonuniform strategies for tableaux calculi for modal logics. *Journal of Applied Nonclassical Logics*, 5(1):77–98, 1995.
5. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of the 13th Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag. Also DIST-Technical Report 96-0037 and IRST-Technical Report 9601-02.
6. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). Technical Report 9611-06, IRST, Trento, Italy, 1996. A shorter version is submitted to "Information and Computation".
7. J. Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(3):361–372, 1995.
8. J.Y. Halpern and Y. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.
9. S. Kirkpatrick and B. Selman. Critical behaviour in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.
10. D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 459–465, 1992.
11. C. P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

This article was processed using the LaTeX macro package with LLNCS style