*SAT, KSATC, DLP and TA: a comparative analysis

Enrico Giunchiglia DIST, V.le Causa 13 16145 Genova, Italy

Fausto Giunchiglia IRST, 38050 Povo, Trento, Italy

Armando Tacchella DIST, V.le Causa 13 16145 Genova, Italy

Abstract

*SAT is our new platform for building decision procedures for modal and description logics. Currently, *SAT can test the satisfiability of a formula in the modal logic K or in the description logic $\mathcal{ALC}$. We comparatively test *SAT with some among the fastest solvers for K: KSATC, DLP and TA. The experimental analysis shows that *SAT performs better than or as well as the other systems on the tests we consider.

1 Introduction

*SAT is our new platform for building decision procedures for modal and description logics. Currently, *SAT can test the satisfiability of a formula in the modal logic K or in the description logic $\mathcal{ALC}$. It is out of the goals of this paper to describe *SAT structure, optimizations and configurable options. For a more detailed presentation, see [1] and the manual distributed with *SAT. We only remark that:

- *SAT is built on top of the SAT decider SATO [2]. SATO is one of the fastest among the currently available SAT-solvers, and has many configurable options. *SAT inherits and exploits the original SATO options, and allows for some more (including early pruning, caching, two forms of backjumping, and the choice of various splitting heuristics); and
- *SAT is designed to support the development of SAT-based deciders for modal and description logics. Currently, *SAT is able to deal with eight modal logics. For some of these logics, we do not know of any other implemented decision procedure, nor of any reduction to a formalism for which a decision procedure is available.

In this paper, we limit ourselves to comparatively test *SAT with other state-of-the art systems on a set of randomly generated tests and on the benchmark formulas for K used at the Comparison of Theorem Provers for Modal Logics at Tableaux'98 (see [3]). The experimental analysis shows that on these tests *SAT performs better than or as well as the other systems.

2 A comparative analysis

In our comparative experimental analysis, we consider the four systems *SAT, KSATC [4], DLP [5] and TA [6], i.e., some among the fastest solvers for K. We remember that TA, given a modal formula φ, first determines a corresponding first order formula φ^* and then it performs conventional first-order theorem proving. In our tests, as in [7], TA uses FLOTTER to convert φ^* in a set of clauses $Cl(\varphi^*)$, and then the theorem prover SPASS to solve $Cl(\varphi^*)$. For a brief description of FLOTTER and SPASS, see [8].[1]

We test these systems on problem sets of randomly generated 3CNF_K formulas. A 3CNF_K formula is a conjunction of 3CNF_K clauses, each with three disjuncts. Each disjunct in a 3CNF_K clause is either a propositional literal or a formula having the form $\Box C$ or $\neg\Box C$, where C is a 3CNF_K clause. See [9] for a more detailed presentation.

Sets of 3CNF_K formulas can be randomly generated according to (*i*) the modal depth d; (*ii*) the number L of clauses at depth $d = 0$; (*iii*) the number N of propositional variables; (*iv*) the probability p with which an atom occurring in a clause at depth $< d$ is purely propositional. Care is taken in order to avoid multiple occurrences of a formula in a clause, at the same time ensuring that the modal vs. the propositional structure of each generated formula only depends on p. In more detail, a clause is generated by randomly generating its disjuncts. When generating a disjunct, we first decide whether it has to be a propositional literal or not. Then a disjunct of the proper type is repeatedly generated as long as it does not occur in the clause generated so far.

In the first tests we consider, a problem set is characterized by N: d and p are fixed to 1 and 0 respectively

[1] The experimental results have been obtained with DLP ver. 3.1, TA ver. 1.4 (SPASS/FLOTTER ver. 0.55), KSATC ver. 1.0, and *SAT ver. 1.2.

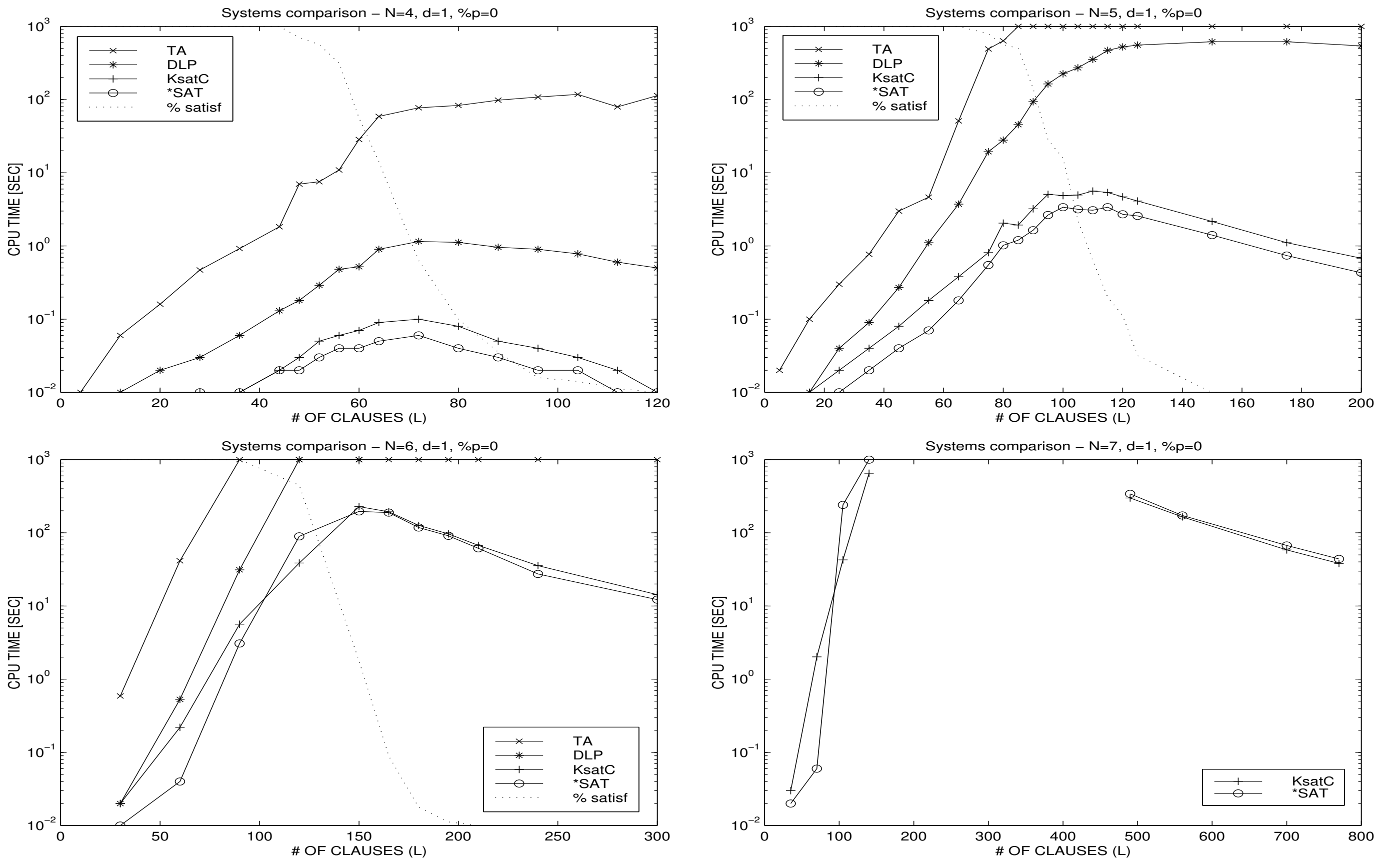

Figure 1: *SAT, KSATC, DLP, and TA median CPU time for $N = 4, 5, 6, 7$. 100 samples/point. Background: satisfiability percentage.

(i.e., the most difficult of the possible settings, see [7]), while L is varied in such a way to empirically cover the "100% satisfiable – 100% unsatisfiable" transition. For each value N in a problem set, 100 $3\mathrm{CNF}_K$ formulas are randomly generated, and the resulting formulas are given in input to the procedure under test. For practical reasons, a timeout mechanism stops the execution of the system on a formula after 1000 seconds of CPU time.

The first three problem sets we consider have $N = 4, 5, 6$ while p is fixed to 0%: according to Hustadt and Schmidt [7], fixing $p = 0\%$ corresponds to particularly difficult tests. We call these problem sets PKN4p0, PKN5p0, and PKN6p0 respectively. PKN4p0 and PKN6p0 are called PS12 and PS13 respectively in [7]. In order to better highlight the behavior of *SAT and KSATC, we also run these systems on a problem set (called PKN7p0) having $N = 7$ and $p = 0\%$. In Figure 1, satisfiability percentages and the median of the CPU times for the four systems are plotted against the number of clauses L. For TA, we only take into account the time needed by SPASS to solve the sample. Notice the logarithmic scale on the vertical axis.

Consider Figure 1. The first observation is that *SAT and KSATC are the fastest. The two systems perform roughly in the same way, with *SAT performing better when $L = 4, 5$. For $L = 6, 7$, the two systems have similar performances, one system performing better than the other for some values of L, but worse for other values of L. This comes at no surprise: the two systems have the same underlying structure, both use early pruning, and in both of them disjuncts are chosen according to their number of occurrences in small clauses (Maximum Occurrences in clauses of Minimum Size, MOMS). *SAT and KSATC do not have an identical behavior because they use different data-structures and implement slightly different MOMS strategies.

Considering the other systems, for PKN4p0 the gap between *SAT/KSATC and DLP [resp. TA] is of more than one order of magnitude at the cross-over point of 50% of satisfiable formulas, and goes up to almost 2 [resp. 4] orders of magnitude at the right end side of the horizontal axis. For PKN5p0 and PKN6p0, TA median values exceed the timeout for $L = 85$ and $L = 90$ respectively, while the corresponding values of *SAT [resp. KSATC] are 1.22 [resp. 1.83] and 3.32 [resp. 5.64] seconds. TA keeps exceeding the timeout for all the succes-

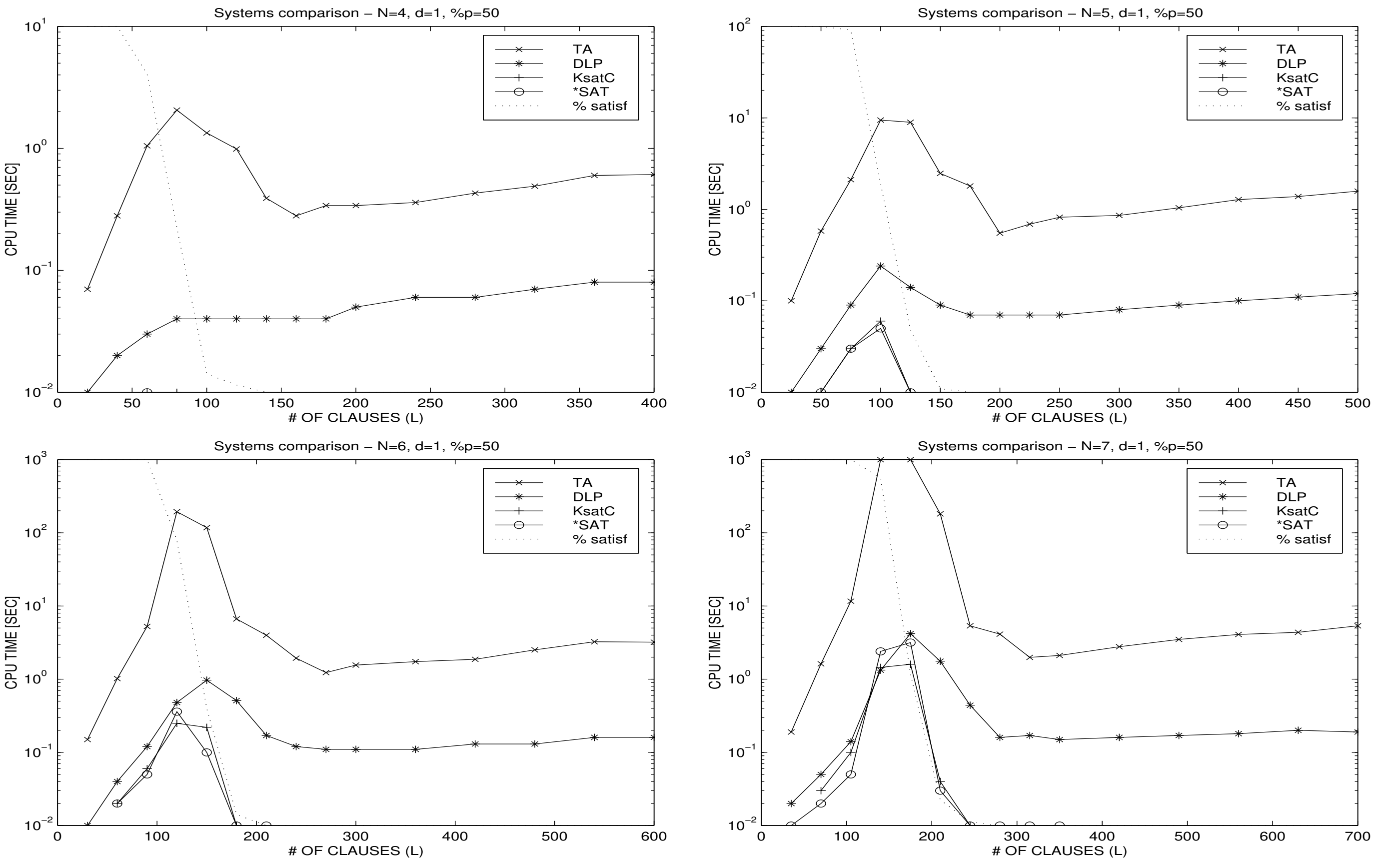

Figure 2: *SAT, KSATC, DLP, and TA median CPU time. $N = 4, 5, 6, 7$. $p = 50\%$. 100 samples/point. Background: satisfiability percentage.

sive values. The gap between *SAT/KSATC and DLP on PKN5p0 is of more than one order of magnitude at the cross-over points of 50% satisfiable formula, and goes up to almost 3 orders at the very right of the plot. When $N = 6$, DLP median values exceed the timeout for $L = 120$ and $L = 150$. For $L > 150$, DLP does not terminate gracefully.[2] Such good performances by *SAT and KSATC are due to early-pruning, which has revealed to be very effective on these problem sets. For example, if we disable early-pruning in *SAT and rerun it on PKN5p0, the system keeps exceeding the time limit for $65 \leq L \leq 200$.

We also run the four systems on problems with $N = 4, 5, 6, 7$ while p is fixed to 50%. We call these problem sets PKN4p50, PKN5p50, PKN6p50, and PKN7p50. In Figure 2 the satisfiability percentages and the median of the CPU times for the four systems are plotted against the number of clauses L. As for $p = 0\%$, *SAT and KSATC perform roughly in the same way and are the fastest. Differently from the tests in which $p = 0\%$, at the transition point of 50% of satisfiable formulas, the gap between *SAT/KSATC and DLP seems to diminish when the number of variables increases. Horrocks and Patel-Schneider [10] show that for values of L bigger than 7, DLP performances are superior to those of KSATC when $d = 1$ and $p = 50\%$. They also conclude that DLP performs better than KSATC when p is high and worse when p is low. We have not yet done such a broad comparison using *SAT instead of KSATC. However, we believe that Horrocks and Patel-Schneider's conclusions should extend also to *SAT, if *SAT is used with the parameter settings we have currently used, i.e., those which make *SAT most similar to KSATC. Of course, a big role can be played by *SAT already available configurable options, and this will be the issue of future research. In any case, both *SAT and KSATC perform better than the other systems for large values of L, when the formulas are trivially unsatisfiable. This is due to the fact that for large values of L, formulas become propositionally unsatisfiable, and thus both *SAT and KSATC mostly take advantage of their SAT-based nature, e.g., of their optimized data structures for handling large formulas.

We observe that for most of the random generated

[2] DLP ver. 3.2 is able to successfully handle formulas with $N = 6$, $p = 0\%$ and $L > 150$.

	*SAT		DLP		TA		
Test	Size	Time	Size	Time	Size	SPASS	FLOTTER
k_branch_n	12	94.49	12	52.29	6	$\underline{84.21}$	12.23
k_branch_p	21	0.21	18	44.56	6	$\underline{51.95}$	13.81
k_d4_n	21	2.87	21	3.43	$\underline{14}$	1.14	42.92
k_d4_p	21	0.06	21	0.17	15	0.64	$\underline{70.47}$
k_dum_n	21	0.12	21	0.13	16	1.75	$\underline{64.07}$
k_dum_p	21	0.04	21	0.07	17	3.32	$\underline{61.67}$
k_grz_n	21	0.01	21	0.20	21	0.16	0.17
k_grz_p	21	0.04	21	0.11	21	0.35	0.16
k_lin_n	21	47.80	21	0.70	21	16.07	63.94
k_lin_p	21	0.01	21	0.08	21	1.03	8.21
k_path_n	21	0.96	21	1.29	4	$\underline{58.70}$	2.14
k_path_p	21	0.72	21	1.19	5	$\underline{22.85}$	2.18
k_ph_n	12	0.60	9	40.16	9	$\underline{45.21}$	9.92
k_ph_p	8	48.54	6	11.34	6	$\underline{42.19}$	0.97
k_poly_n	21	2.25	21	0.65	4	1.23	$\underline{7.86}$
k_poly_p	21	1.73	21	0.34	5	2.48	$\underline{51.00}$
k_t4p_n	21	1.28	21	0.45	9	3.37	$\underline{84.35}$
k_t4p_p	21	0.29	21	0.21	16	3.91	$\underline{84.75}$

Table 1: *SAT, DLP and TA performances on Tableaux'98 benchmarks

problem sets we consider, all the systems seem to have an easy-hard-easy pattern, whose peak roughly correspond to the 50% of satisfiable formulas. When $p = 0\%$, this phenomenon is best evident for *SAT and KSATC.

We also consider the benchmarks formulas for K used at the Comparison of Theorem Provers for Modal Logics at Tableaux'98 (see [3]). These consist of nine provable parameterized formulas (ending with "_p") and nine unprovable parameterized formulas (ending with "_n"). For each parameterized formula $A(n)$, the test consists in determining the greatest natural number $n \leq 21$ satisfying the following two conditions:

1. the prover returns the correct result for the formulas $A(1), A(2), \ldots, A(n)$ in less than 100 seconds, and
2. the prover cannot do the formula $A(n+1)$ in less than 100 seconds or $n = 21$.

Even though it has been proved that most of these tests can be easily solved by current solvers, these are still interesting because

- they are not $3\mathrm{CNF}_K$ formulas, and
- some of these tests have not been solved yet.

The results for *SAT, TA and DLP are reported in Table 1. KSATC has not been tested since KSATC is able to deal with $3\mathrm{CNF}_K$ formulas only. We also show the CPU time requested by the system to solve the last instance $A(n)$. For TA, we do not take into account the time needed to compute the first order formula $A^*(n)$ corresponding to $A(n)$ (which is negligible), but we do take into account the time requested by FLOTTER to convert $A^*(n)$ into a set $Cl(A^*(n))$ of clauses (reported in the FLOTTER column), and the time requested by SPASS to determine the consistency or inconsistency of $Cl(A^*(n))$ (reported in the SPASS column). Furthermore, we stopped TA on $A(n)$ with $n < 21$, either because FLOTTER does not terminate gracefully when computing $Cl(A^*(n+1))$, or because SPASS or FLOTTER exceed the 100 seconds time limit. In the table, these three cases correspond to the rows in which the value for n/SPASS/FLOTTER respectively is underlined.

As can be observed from Table 1, the three systems are able to solve all the instances of a formula in four cases. *SAT and DLP are able to solve all the instances except for k_branch_n, k_branch_p, k_ph_n, k_ph_p. Except for the first of these four parameterized formulas, *SAT is able to solve more instances than DLP. For k_branch_n, both *SAT and DLP are able to solve the 12th instance, with DLP taking less time than *SAT to solve it.

3 Conclusions and future work

In this paper we have compared *SAT performances with respect to other state-of-the-art systems on sets of randomly generated tests and on the benchmarks for K used at the Comparison of Theorem Provers for Modal Logics at Tableaux'98. On these tests, *SAT results are very positive. In a much broader experimental comparison involving KSATC and DLP, Patel-Schneider and Horrocks [10] reach similar conclusions on the random tests we consider, at the same time showing that DLP performs much better than KSATC on tests generated with a different choice for the parameters.

For the future, we plan to extend our work in several directions. We will conduct an extensive experimental analysis (similar to that presented in [11, 12]) to understand, for each class of formulas, which combination of *SAT options leads to the best results. We will enhance *SAT by tuning the existing optimization and by incorporating some new ones. Finally, we will extend *SAT to expressive decidable modal and description logics. Up to date information about our work is available on the WWW at the *SAT project homepage:

`http://www.mrg.dist.unige.it/~tac/StarSAT.html`

Acknowledgements

We are grateful to Hullrich Hustadt, Peter F. Patel-Schneider, and Hantao Zhang for the help they provided on their systems. Special thanks to Roberto Sebastiani for useful discussions related to the subject of this paper.

References

[1] Armando Tacchella. *SAT system description. In *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.

[2] H. Zhang. SATO: An efficient propositional prover. In William McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, July13–17 1997. Springer.

[3] H. de Swart, editor. *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence. Springer-Verlag, May 1998.

[4] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[5] P. F. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 87–89. CEUR, May 1998.

[6] U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[7] U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. Research report MPI-I-97-2-003, Max-Planck-Institut für Informatik, Saarbrücken, Germany, February 1997.

[8] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER version 0.42. In M.A. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th Conference on Automated Deduction (CADE-13)*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 141–145, New Brunswick, New Jersey, USA, July/August 1996. Springer.

[9] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. CADE-96*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag.

[10] I. Horrocks and P. F. Patel-Schneider. Advances in propositional modal satisfiability, 1999. Manuscript.

[11] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 90–94. CEUR, May 1998.

[12] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 1999. To appear.