The Random Modal QBF Test Set

Juan Heguiabehere and Maarten de Rijke

Institute for Logic, Language and Computation (ILLC)
Faculty of Science, University of Amsterdam
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands
E-mail: {juanh,mdr}@science.uva.nl

Abstract. We provide an empirical evaluation of one of the main test sets that is currently in use for testing modal satisfiability solvers, viz. the random modal QBF test set. We first discuss some of the background underlying the test set, and then evaluate the test set using criteria set forth by Horrocks, Patel-Schneider, and Sebastiani. We also present some guidelines for the use of the test set.

1 Introduction

Usually, theoretical studies do not provide an indication of the effectiveness and behavior of complex systems such as satisfiability solvers. Instead, empirical evaluations have to be used. In the area of propositional satisfiability checking there is large and rapidly expanding body of experimental knowledge; see e.g., [3]. In contrast, empirical aspects of modal satisfiability checking have only recently drawn the attention of researchers. We now have a number of test sets, some of which have been evaluated extensively [2, 5, 4, 7, 6]. In addition, we also have a clear set of guidelines for performing empirical testing in the setting of modal logic; these were proposed by Horrocks, Patel-Schneider, and Sebastiani [6], building on work by Heuerding and Schwendimann [5].

The aim of this paper is to provide an empirical evaluation of one of the test sets that is currently in use for testing modal satisfiability solvers, viz. the random modal QBF test set. We start by discussing the modal QBF test set and some of the underlying intuitions. We then evaluate the test set using criteria set forth by Horrocks, Patel-Schneider, and Sebastiani. We also present some guidelines for the use of the test set.

2 The Random Modal QBF Test Set

The first random generation technique used in testing modal decision procedures, the random $3\mathrm{CNF}_{\Box_m}$ test methodology, was proposed in [4]; its subsequent development is described in [6]. The random modal quantified Boolean formula (QBF) test set was proposed by Massacci [9], and used in the 1999 and 2000 editions of the TANCS system comparisons [13]. It is based on the idea of randomly generating QBFs and then translating these into modal logic. Let us explain these two steps in more detail.

Generating QBFs

Recall that QBFs have the following shape: $\mathsf{Q}_1 v_1 \ldots \mathsf{Q}_n v_n \; CNF(v_1, \ldots, v_n)$. That is, QBFs are prenex formulas built up from proposition letters, using the booleans, and $\forall v\, \beta$ and $\exists v\, \beta$ (where v is any proposition letter).

What is involved in evaluating a QBF? We start by peeling off the outermost quantifier; if it's $\exists v$, we choose one of the truth values 1 or 0 and substitute for the newly freed occurrence of v; if it's $\forall v$, substitute both 1 and 0 for the newly freed occurrences of v. In short, while evaluating QBFs we are generating a tree, where existential quantifiers increase the depth, and universal quantifiers force branching.

In the *random modal QBF test set*, 4 parameters play a role: c, d, v, k:

- The parameter c is the number of clauses of the randomly generated QBF.
- The parameter d is the alternation depth of the randomly generated QBF; it is *not* the modal depth of the modal translation. (More on this below.)
- The parameter v is the number of variables used per alternation.
- And k is the number of different variables used per clause.

The *QBF-validity problem* is the problem of deciding whether a QBF without free variables is valid; it is known to be PSPACE-complete. For every fixed valued of d we can capture the problems in Σ_d^P in the polynomial hierarchy; PSPACE can only be reached by an unbounded value of d.

Here's a concrete example. Using $d = 3$ and $v = 4$ we can generate

$$\underbrace{\forall \overbrace{v_{34}v_{33}v_{32}v_{31}}^{4} \exists \overbrace{v_{24}v_{23}v_{22}v_{21}}^{4} \forall \overbrace{v_{14}v_{13}v_{12}v_{11}}^{4} \exists \overbrace{v_{04}v_{03}v_{02}v_{01}}^{4}}_{3} \mathit{CNF}(v_{01}, \ldots, v_{34}).$$

Each clause in $\mathit{CNF}(v_{01}, \ldots, v_{34})$ has k different variables (default 4) and each is negated with probability *prneg* (default 0.5). The first and the third variable (if it exists) are existentially quantified. The second and fourth variable are universally quantified. This aims at eliminating trivially unsatisfiable formulas. Other literals are either universal or existentially quantified variables with probability *prmod* (default 0.5). The depth of each literal is randomly chosen from 1 to d.

By increasing the parameter d from odd to even, a layer of existential quantifiers is added at the beginning of the formula, and, conversely, when d increases from even to odd, a layer of universal quantifiers is added. The impact of increasing either v or d on the shape of the QBF trees may be visualized as in Figure 1, beginning from the smallest case where $v = 2$ and $d = 1$.

Translating into Modal Logic

The QBF that is produced by the random generator is translated into the basic modal logic with the usual boolean operators and $\Box$, $\Diamond$, using a variant of an encoding that is originally due to Ladner [8]. The core idea underlying the translation is to capture, by means of a modal formula, the 'peel off quantifiers and substitute' evaluation process for a given input QBF. The translation forces branching in the structure of the possible model whenever an universal quantifier is found in the original formula, keeps the branches separate, and makes sure there are enough modal levels in the model. It forces the structure of the possible model to be a tree, and the resulting formula is satisfiable iff the original formula is.

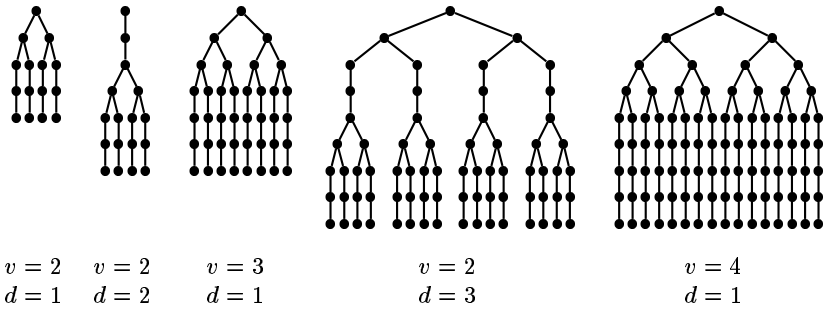


Fig. 1. The shape of QBF trees.

Here's a detailed example. The formula $\phi = \forall v_4 v_3 \exists v_2 v_1 \, (v_1 \vee \neg v_2 \vee \neg v_3 \vee \neg v_4)$ (generated with parameters $v = 2$, $d = 1$, $c = 1$, default encoding) translates into the conjunction of the following formulas.

- The matrix ϕ must be true everywhere in the model at the depths in which its variables become fixed in value: $\bigwedge_{m=1}^{4} \Box^m (v_1 \vee \neg v_2 \vee \neg v_3 \vee \neg v_4)$, where $\Box^m$ is a sequence of m occurrences of the $\Box$ operators.
- Keep values of proposition letters forever, adding one a level, in order of quantifier appearance:
  - $\bigwedge_{m=1}^{3} (\Box^m (v_4 \vee \Box(\neg v_4)) \wedge \Box^m (\neg v_4 \vee \Box(v_4))) \wedge$
  - $\bigwedge_{m=1}^{2} (\Box^m (\Box(v_3 \vee \Box(\neg v_3))) \wedge \Box^m (\Box(\neg v_3 \vee \Box(v_3)))) \wedge$
  - $\Box(\Box(\Box(v_2 \vee \Box(\neg v_2)))) \;\wedge\; \Box(\Box(\Box(\neg v_2 \vee \Box(v_2))))$
- Force branching on universal quantifiers: $\Diamond v_4 \wedge \Diamond \neg v_4 \wedge \Box(\Diamond v_3) \wedge \Box(\Diamond \neg v_3)$.
- Force tree depth; note that the first two levels are covered by the previous two formulas: $\Box(\Box(\Diamond(\top)))) \wedge \Box(\Box(\Box(\Diamond(\top)))))$.

During the translation formulas are added to guarantee the alternation of quantifiers in a tree-like form.

The parameters c, k, v and d that are used in the generation process are related to the final modal formula in the following way. The (maximum) number of clauses is $c \cdot k + (v \cdot (d+1))^2 + \lfloor v \cdot (d+1)/2 \rfloor$. The (maximum) number of proposition letters is $v \cdot (d+1)$. And the (maximum) modal depth is $v \cdot (d+1)$. These maximums obtain when c is high enough compared to $v \cdot (d+1)$ to cover all the possible proposition letters. The file size for the translated formula is linear in c, and polynomial in v and d, but usually we are not interested in big values of the last two, so this is not much of a problem.

3 Test Results

Settings

To evaluate the QBF test set, we used 3 satisfiability solvers for modal logic. First, we used the general first-order prover SPASS [11], version 1.0.3, extended with the layered

translation of modal formulas into first-order formulas as presented in [1]. Second, we used MSPASS version V 1.0.0t.1.2.a [10]. And, third, we used *SAT version 1.3 [12].

Our experiments were run on a Pentium III 800 MHz with 128 MB of memory, running RedHat Linux 7.0.

Juan, please supply version number

We used version XXXX of the QBF problem generator. To facilitate future comparisons, we used as many default settings as possible: no modal encoding, no lean encoding, the number of literals set to 4, and *prmod* and *prneg* both set to 0.5. We generated 64 instances of each problem, and the outputs of the generator were translated to the formats of the provers being used; in one case we had to convert modal formulas to first-order logic formulas. The resulting file sizes were linear in c, even though the linear coefficient varied from one solver to another.

Our main measurements concerned both CPU time elapsed (with a 10800 second timeout) and a time independent measure: the number of clauses generated for SPASS plus layering and for MSPASS, and the number of unit propagations for *SAT.[1]

Findings

We first ran the standardized tests provided by the TANCS competition: 64 instances randomly generated with $c = 20$, $v = 2$, $d = 2$, and default settings for the remaining parameters. See Figure 2.

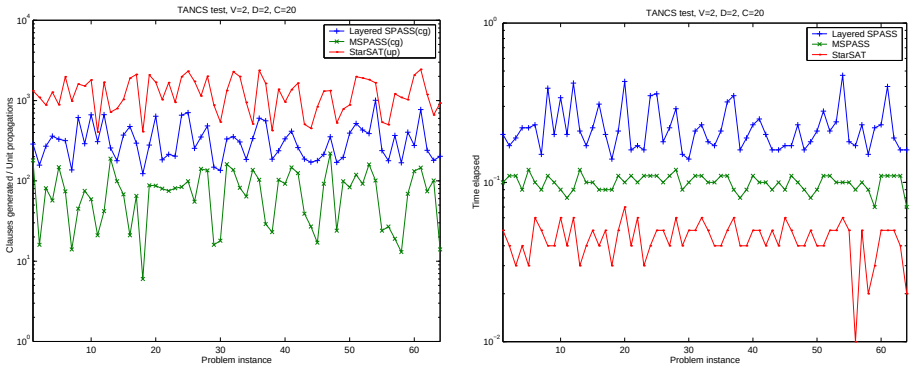


Fig. 2. The standardized tests provided by TANCS, used for SPASS, MSPASS, and *SAT. (Left): clauses generated/unit propagations per problem instance, log scale. (Right): CPU time (seconds) per problem instance, log scale.

While the number of clauses generated by resolution provers and the number of unit propagations in *SAT are not directly comparable as a performance measurement, they do give an indication of the relative difficulty of a problem (or problem set). As such, we can see that the difficulty of a problem varies with the method used to solve it. The

[1] We found that the number of unit propagations best describes the resource usage for each problem, as evidenced by its strong correlation to elapsed time. Unfortunately, a detailed evaluation of unit propagations versus other statistics reported by *SAT (such as assignments found) as a suitable time independent measure, is beyond the scope of this paper.

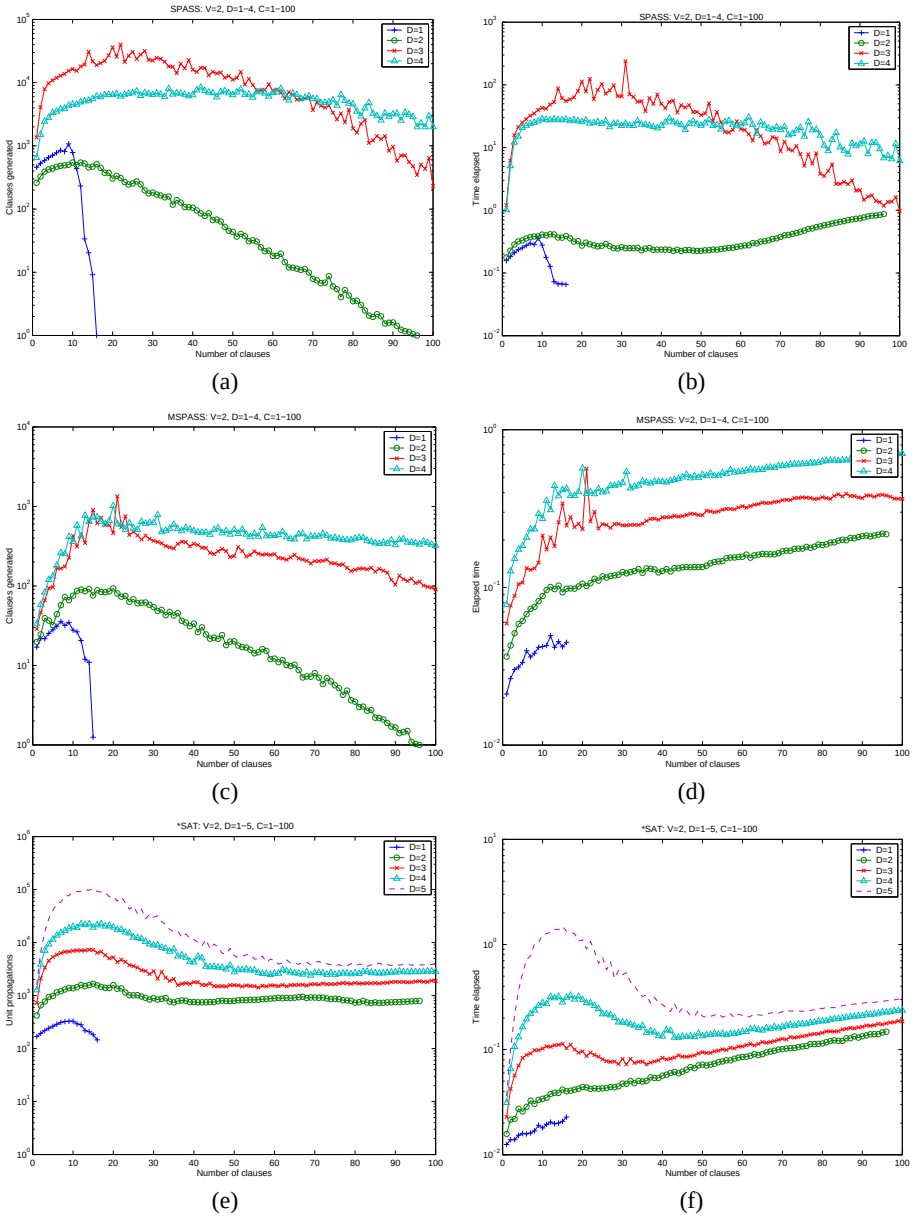


Fig. 3. SPASS, MSPASS, and *SAT on QBF test sets, $v = 2$, $d = 1$–4 (5), 64 samples/point, mean values. (Left): clauses generated/unit propagations, log scale. (Right): CPU time in seconds, log scale.

correlation between time elapsed and clauses generated/unit propagations varies widely between the methods. In fact, for this test the *SAT times are completely dominated by startup costs and don't really inform us about problem difficulty.

Next we ran a number of sweeps, with each of three provers, with $v = 2$ and increasing d from 1 to 4 (and to 5 in the case of *SAT), while increasing c from 1 to 100. The resulting CPU times and the number of clauses generated/unit propagations are depicted in Figure 3; the curves for $d = 1$, $d = 2$ do not extend to the right-hand side of the plots, as the formulas being generated with these settings are simply too small to be be able to accommodate larger number of clauses.

Several things are worth noting about Figure 3. First, the sets display an easy-hard-easy pattern familiar from propositional satisfiability testing [3]. The shape of the curves is strongly dependent on the solver used. Moreover, the pattern seem to vary from not-too-hard-hard-easy in some cases (SPASS, $d = 1$, $d = 2$, $d = 4$) to not-too-hard-hard-hard in others (SPASS, $d = 4$; MSPASS, $d = 3$, $d = 4$) to not-too-hard-hard-not-too-hard in yet others (SPASS, $d = 3$; *SAT, $d = 2$, $d = 3$, $d = 4$, $d = 5$).

Second, for both SPASS and MSPASS we see that curves cross each other; this is most clearly visible in (a), where the number of clauses generated by SPASS are displayed, but it also shows up in (b) where the CPU times for SPASS are shown. Hence, for SPASS (and to a lesser extent for MSPASS) the d parameter does not influence the difficulty of the problems being generated in a monotonic way.

Third, the time elapsed (displayed in (b), (d), and (f)) has a very strong dependence on file size: after the hard region has been crossed and the elapsed time tends to decrease, it actually starts going up again. The impact of input file size and I/O is most noticeable for MSPASS (plot (d)); but even in the case of *SAT, where the number of unit propagations remains more or less constant after the hard region has been traversed, the CPU times start going up: this increase is entirely due to input file size and I/O. In Figure 4 we have plotted the growth of the input file size against c, and against d. The file size can be approximated by $11000 + c * 485$, while the preprocessing performed by the layered translation brings this up to $20000 + c * 930$. Remarkably, the translated file for MSPASS is *smaller* than the original input file.

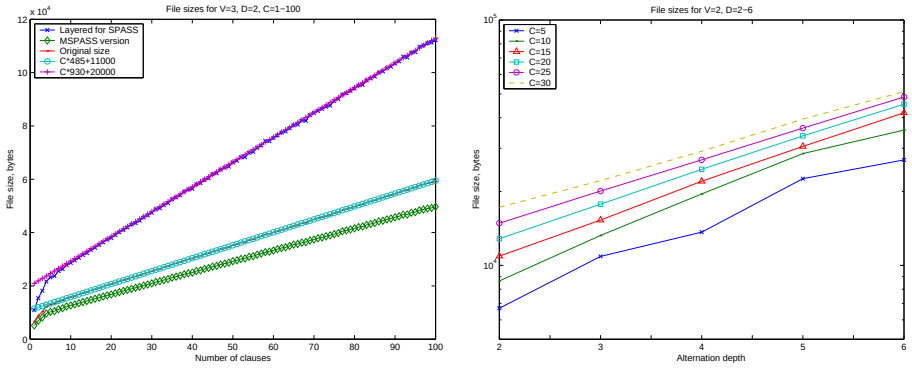


Fig. 4. Size of the input files, 64 samples/point, mean values. (Left): as a function of the number of clauses. (Right): as a function of the alternation depth, original size.

When we increased the parameter v, we saw similar curve shapes as for $v = 2$. In Figure 5 we have displayed the results of running *SAT with $v = 3$. Notice that the humps indicating the hard regions are higher for $v = 3$ than for $v = 2$ (see Figure 3

(e) and (f)), indicating that the problems are harder; hence, the CPU times are not as strongly dominated by file size and I/O aspects as in the case where $v = 2$. The fact that the hard regions are 'wider' than for $v = 2$ indicates that we are not only getting harder problems, but also that the fraction of hard problems is increasing.

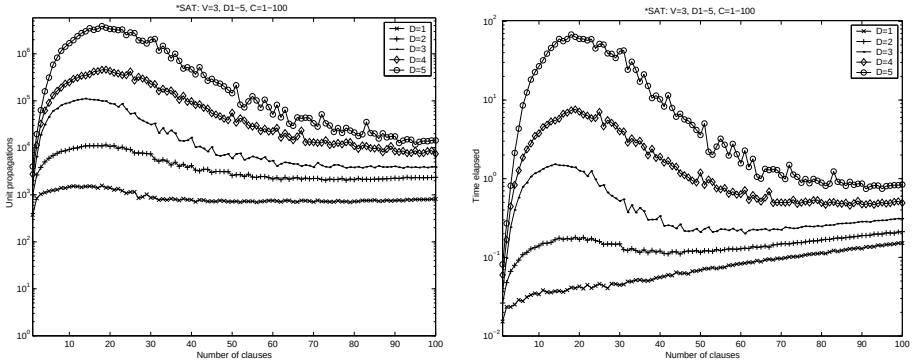
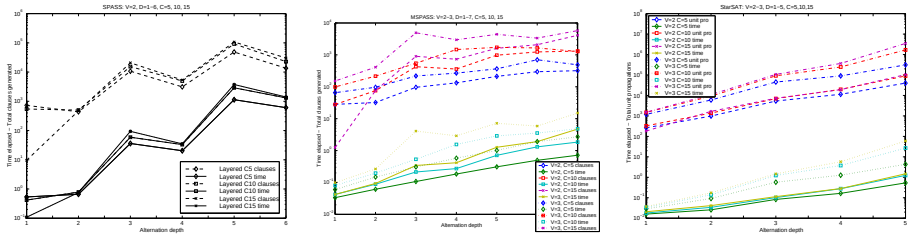


Fig. 5. *SAT results for $v = 3$, $d = 1$–5, 64 samples/point, mean values (Left): unit propagations, log scale. (Right): CPU time in seconds, log scale.

Let us return to the phenomenon observed in Figure 3, where it was found that the d parameter does not monotonically control difficulty. We can observe this even clearer when we plot d along the x-axis, as in Figure 6. Note that the phenomenon is strongly prover dependent: it clearly shows up for SPASS (with the layered translation) as shown in (a); it is somewhat visible with MSPASS (b), but not at all with *SAT (c). Further experimental work has shown that this 'staircase phenomenon' is also present with larger values of v for SPASS. The phenomenon is related to the special way in which QBFs grow: existential quantifiers are added to the original QBF when d is increased from odd to even, universal quantifiers when d is increased from even to odd; see Figure 1. The former simplifies matters for SPASS with the layered translation, while the latter make matters considerably harder for that solver.[2]



(a) (b) (c)

Fig. 6. SPASS, MSPASS and *SAT results for $v = 2$ (3), $d = 1$–6 (7), 64 samples/point, mean values. (a): SPASS with $v = 2$. (b): MSPASS with $v = 2, 3$. (c): *SAT with $v = 2, 3$. Log scales.

[2] Note that the staircase phenomenon will not be observed if one only performs the standardized TANCS test as this test only involves a single value of d.

One central concern with any test set, synthetic or not, is *parametrization*: to which extent can we choose the difficulty of the problem and of exploring the input space? In the QBF test set the difficulty can easily be controlled: the v parameter controls it monotonically, the d parameter also with some caveats. It seems, however, that v and d do not control truly *independent* dimensions of the problem space. More precisely, combinations of v and d for which the value of $v \cdot (d+1)$ coincides have very similar curves, as can be seen in Figure 7. This suggests that $v \cdot (d+1)$ is the dimension along which the QBF problem space should be explored, instead of either v or d independently. This comes as no surprise: as we saw in Section 2, $v \cdot (d+1)$ determines both the maximum number of propositional variables and the maximum modal depth in the translated formula. (As an aside, it is clear from Figure 7 that with increasing values of $v \cdot (d+1)$, the truly hard region for a given setting of parameters moves to the right as we increase the number of clauses.)

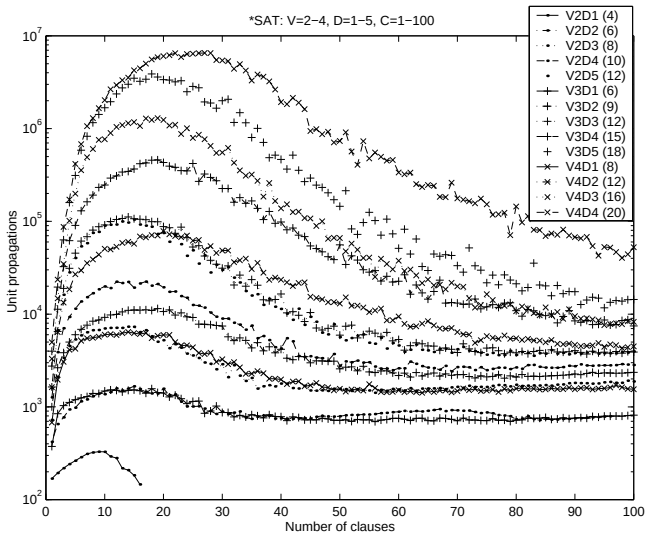


Fig. 7. *SAT results for $v = 2$–4, $d = 1$–5, and $c = 1$–100, 64 samples/point, mean values. The numbers in brackets indicate the value of $v \cdot (d+1)$.

An important aspect that we have not discussed so far is the satisfiable vs. non-satisfiable fraction. The parameter c does indeed allow us to control the satisfiability fraction: it goes from 1 to 0 monotonically with c. However, there are remarkably few values of c for which the satisfiable fraction is 1; see Figure 8. As illustrated by Figure 8 (top left), we have found satisfiable fractions of about 20% in many repeated runs of the standardized 20/2/2 TANCS test (see Figure 2). Moreover, there is a heavy 'tail' of unsatisfiable problems, as indicated by the curves in Figure 7. And contrary to intuition, the constrainedness of problems does not seem to depend very strongly on the d parameter; for a fixed v, increasing d from odd to even doesn't shift the satisfiable fraction graph by any noticeable amount. The constrainedness of the underlying models, then, remains unchanged despite the addition of variables and the increase in depth.

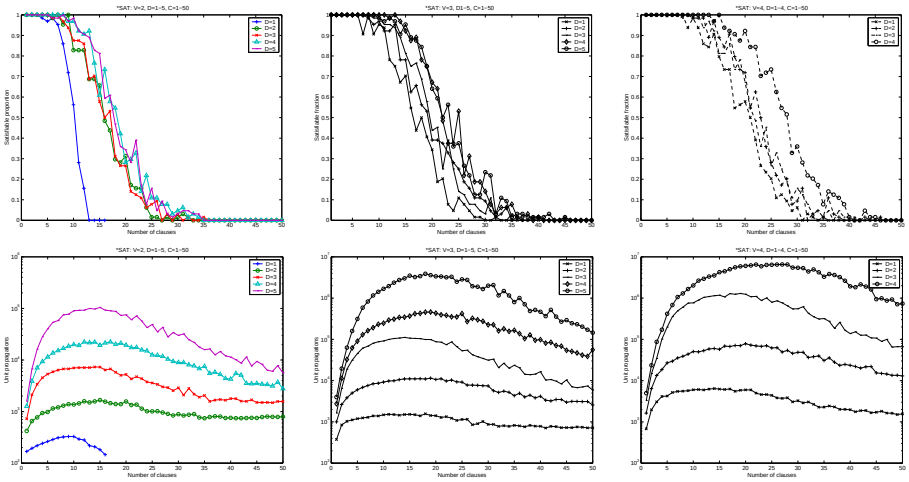


Fig. 8. *SAT results for $d = 1$–4 (5), $c = 1$–50, 64 samples/point, mean values. (Top): Satisfiable fraction. (Bottom): Unit propagations. (Left): $v = 2$. (Middle): $v = 3$. (Right): $v = 4$.

There's one more thing that's worth point out. The 50% satisfiable mark occurs towards the right-hand side of the truly hard region; this may suggest that the hardest problems within a given test set are likely to be satisfiable ones.

Finally, recall that a modal formula is *trivially satisfiable* iff it is satisfiable on a model with a single node [6, 7]. Clearly, trivial satisfiability is not a problem for modal QBF test sets. For a start, very few satisfiable problems are generated anyway, and because of the highly structured form of the randomly generated QBFs, the resulting modal formulas always contain $\Diamond$-subformulas, thus avoiding trivial satisfiability.

4 Discussion and Conclusion

The general criteria put forward by Horrocks, Patel-Schneider, and Sebastiani [6] for evaluating modal test methodologies, boil down to demanding a reproducible sample of an interesting portion of the input space with appropriate difficulty. To conclude this paper, here's a brief discussion of these criteria as they relate to our setting.

Obviously, *reproducibility* is guaranteed for the modal QBF test set. The modal QBF test set seems to represent just a restricted area of the whole input space; that is, it scores low on *representativeness*. There are three reasons for this. First, the QBF test set provides poor coverage of the satisfiable region, and especially of the easily satisfiable region; most of the modally encoded QBF-formulas generated with values of v and d that are within reach of today's tools, are hard and unsatisfiable, as suggested by Figure 8. Second, the modally encoded QBFs are of a very special shape, which seems to lead to the so-called staircase phenomenon for some solvers. And third, the v and d parameters end up being substantially overlapping and interrelated as part of the translation of QBFs into modal formulas. A strong point in favor of the QBF test set is that it is possible to generate hard problems with a large modal depth which are still

within reach of today's modal satisfiability solvers; in this respect the QBF random test methodology fares better than the New_3CNF$_{\Box_m}$ test methodology, as reported in [6].

The QBF test set scores low on the *satisfiable vs. unsatisfiable balance*: it has a strong preference for generating unsatisfiable instances. The levels of *difficulty* offered by the test set are sufficient, as they range from next to trivial to too hard for today's systems. The tests *terminate* and provide information in a reasonable amount of time. Finally, the set suffers from *over-size*, which forces one to consider both time dependent and time independent measurements for determining the performance of solvers.

In conclusion, then, the random modal QBF test methodology provides useful test sets that should, however, not be used as the sole measure in the evaluation of modal satisfiability solvers. In particular, the standardized tests provided by TANCS (with $c = 20$ and $v = d = 2$) does not provide informative measurements.

Plans for further work include explorations beyond the default settings (more literals, modal and/or lean encodings, alternative settings for $prneg$, $prmod$, ...), comparisons with QBF solvers, and systematic comparisons with other test methodologies.

Acknowledgments. We would like to thank our referees for valuable comments and suggestions. Both authors were supported by the Spinoza project 'Logic in Action.' Maarten de Rijke was also supported by a grant from the Netherlands Organization for Scientific Research (NWO) under project number 365-20-005.

References

1. C. Areces, R. Gennari, J. Heguiabehere, and M. de Rijke. Tree-based heuristics in modal theorem proving. In W. Horn, editor, *Proceedings ECAI 2000*, 2000.
2. 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. In *Proceedings KR-92*, 1992.
3. I. Gent, H. van Maaren, and T. Walsh, editors. *SAT 2000*. IOS Press, 2000.
4. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K. In *Proceedings CADE-96*, 1996.
5. A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, and S4. Technical report IAM-96-015, University of Bern, Switzerland, 1996.
6. I. Horrocks, P.F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8:293–323, 2000.
7. U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proceedings IJCAI-97*, pages 202–207, 1997.
8. R. Ladner. The computational complexity of provability in systems of modal logic. *SIAM Journal on Computing*, 6:467–480, 1977.
9. F. Massacci. Design and results of the Tableaux-99 non-classical (modal) system competition. In *Proceedings Tableaux'99*, 1999.
10. MSPASS V 1.0.0t.1.2.a. URL: `http://www.cs.man.ac.uk/~schmidt/mspass`. Accessed February 23, 2001.
11. SPASS Version 1.0.3. URL: `http://spass.mpi-sb.mpg.de/`. Accessed May 23, 2000.
12. A. Tacchella. *SAT system description. In *Proceedings DL'99*, 1999.
13. TANCS: Tableaux Non-Classical Systems Comparison. `http://www.dis.uniroma1.it/~tancs`. Site accessed on January 17, 2000.