The SAT-Based Approach for Classical Modal Logics*

Enrico Giunchiglia[1], Fausto Giunchiglia[2], and Armando Tacchella[1]

[1] DIST, Università di Genova, Viale Causa, 13 – 16145 Genova, Italy
[2] DISA, Università di Trento. IRST, 38050 Povo, Trento, Italy.

Abstract. We present a set of SAT-based decision procedures for various classical modal logics. The decision procedures are implemented in our system *SAT. For some of the logics we deal with, we are not aware of any other implementation. For the others, we define a testing methodology which generalizes the $3CNF_K$ methodology by Giunchiglia and Sebastiani. The experimental evaluation shows that our decision procedures perform better than or as well as other state-of-the-art decision procedures.

1 Introduction

In recent years there has been a growing interest in efficient decision procedures for modal logics. Researchers tried to push performances either by building optimized version of tableau algorithms, or by finding reductions to logics for which state-of-the-art systems are already available. The SAT-based approach, first suggested by Giunchiglia and Sebastiani [1], features yet a different recipe: (*i*) take out off the shelf one of the fastest SAT procedures available, and (*ii*) use it as the basis for your modal decider.

In this paper we present a set of SAT-based decision procedures for various classical modal logics. We have implemented the procedures in our system *SAT which improves on previous works by Sebastiani and ourselves [2, 3, 1]. *SAT is built on top of the SAT decider SATO [4] one of the fastest and better engineered among the publicly available SAT solvers. *SAT is able to deal with eight modal logics and, 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 [5], we experimentally tested *SAT performances on various benchmarks for K against other state-of-the-art deciders. Here, we define a testing methodology which generalizes the $3CNF_K$ methodology by Giunchiglia and Sebastiani [2] and we test *SAT performances in the modal logic E. The experimental analysis shows that *SAT performs better than or as well as other state-of-the-art systems.

2 Classical modal logics

Following [6], a *modal logic* is a set of formulas (called *theorems*) closed under tautological consequence. Most modal logics are closed under the rule

$$\frac{(\varphi_1 \wedge \ldots \wedge \varphi_n) \supset \psi}{(\Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n) \supset \Box\psi}.$$

* We are grateful to Ullrich Hustadt, Peter F. Patel-Schneider, and Hantao Zhang for the assistance they provided on their systems. Thanks to Roberto Sebastiani for useful discussions related to the subject of this paper.

for certain values of n. If $n = 1$ then the logic is said to be *monotone*. If $n = 2$ then the logic is said to be *regular*. If $n \geq 0$ then the logic is said to be *normal*. The smallest monotone, regular and normal modal logics are called M, R and K respectively (see, e.g., [6]).

Classical modal logics [7, 8] are weaker than normal modal logics. In fact, the only requirement is that the set of theorems is closed under the rule

$$\frac{\varphi \equiv \psi}{\Box\varphi \equiv \Box\psi}.$$

As a consequence, the schemas

N: $\Box\top$,[1]
M: $\Box(\varphi \wedge \psi) \supset \Box\varphi$,
C: $(\Box\varphi \wedge \Box\psi) \supset \Box(\varphi \wedge \psi)$,

which are theorems in K do not necessarily hold in classical modal logics. The three principles N, M, and C enforce closure conditions on the set of provable formulas which are not always desirable, especially if the $\Box$ operator has an epistemic (such as knowledge or belief) reading. If we interpret $\Box\varphi$ as "a certain agent a believes φ", then N enforces that a believes all the logical truths, M that a's beliefs are closed under logical consequence, and C that a's beliefs are closed under conjunction. These three closure properties are different forms of omniscience, and —as such— they are not appropriate for modeling the beliefs of a real agent (see, e.g., [9] Chapter 9). We can easily imagine situations involving a's beliefs, where only an arbitrary subset of the above properties holds.

There are eight possible ways to add the three schemas M, C, and N to the smallest classical modal logic E. The resulting modal logics are called E, EM (equivalent to the logic M), EN, EMN, EC, EMC (equivalent to R), ECN, EMCN (equivalent to K), where EX denotes the logic obtained adding the schemas in X to E.

3 SAT-based procedures for classical modal logics

We say that a conjunction μ of propositional literals and formulas of the form $\Box\varphi$ or $\neg\Box\varphi$ is an *assignment* if, for any pair ψ, ψ' of conjuncts in μ, it is not the case that $\psi = \neg\psi'$. An assignment μ *satisfies* a formula φ if μ entails φ by propositional reasoning. A formula φ is *consistent* in a logic L (or L-*consistent*) if $\neg\varphi$ is not a theorem of L, i.e., if $\neg\varphi \notin$ L.

Consider a formula φ. Let S be a set of assignments satisfying φ, and let L be a modal logic. As noticed by Sebastiani and Giunchiglia [10], the following two facts hold:

- If at least one assignment in S is L-consistent then φ is L-consistent.
- If no assignment in S is L-consistent then φ is not L-consistent as long as the set S is *complete for* φ, i.e., as long as the disjunction of the assignments in S is propositionally equivalent to φ.

From these facts, it follows that the problem of determining whether φ is L-consistent can be decomposed in two steps:

- *generate* an assignment μ satisfying φ, and
- *test* whether μ is L-consistent.

[1] The symbols $\top$ and $\bot$ are 0-ary connectives representing truth and falsity respectively.

```
function LSAT(φ)
   return LSAT_DP(cnf(φ), ⊤).

function LSAT_DP(φ, μ)
   if φ = ⊤ then return LCONSIST(μ);          /* base */
   if φ = ⊥ then return False;                /* backtrack */
   if { a unit clause (l) occurs in φ }       /* unit */
      then return LSAT_DP(assign(l, φ),μ ∧ l);
   l := choose-literal(φ, μ);
   return LSAT_DP(assign(l, φ),μ ∧ l) or      /* split */
          LSAT_DP(assign(l̄, φ),μ ∧ l̄).
```

Fig. 1. LSAT and LSAT_{DP}

In all the logics we consider, testing the consistency of an assignment μ amounts to determining the consistency of other formulas whose depth (i.e., the maximum number of nested $\Box$ operators) is strictly minor than the depth of μ. This implies that we can check the consistency of these other formulas by recursively applying the above methodology, at the same time ensuring the termination of the overall process. The above methodology can be implemented by two mutually recursive procedures:

- LSAT(φ) for the generation of assignments satisfying φ, and
- LCONSIST(μ) for testing the L-consistency of each generated assignment μ.

3.1 Lsat(φ)

Consider a formula φ. Let L be a modal logic. The generation of assignments satisfying φ is independent of the particular logic L being considered. Furthermore, it can be based on any procedure for SAT:

- if the SAT decider is complete, then we can generate a finite and complete set of assignments for φ as follows:
  - at step 0, ask for an assignment satisfying φ, and
  - at step $i + 1$, ask for an assignment satisfying φ and the negation of the assignments generated in the previous steps.

  By checking the L-consistency of each assignment, we obtain a correct and complete decider for L.
- if the SAT decider is correct but incomplete, then we cannot generate a complete set of assignments for φ, but we can still build a correct but incomplete decider for L by checking each generated assignment. Of course, whether an incomplete effective procedure for SAT can be turned into an effective incomplete procedure for a modal logic L, is still an open point.

The above method for generating a complete set of assignments for φ has the advantage that the SAT decider is used as a blackbox. The obvious disadvantage is that the size of the input formula checked by the SAT solver may become exponentially bigger than the original one. A better solution is to invoke the test for L-consistency *inside* the SAT procedure whenever an assignment satisfying the input formula is found. In the case of the Davis-Putnam (DP) procedure [11], we get the procedure LSAT represented in Figure 1. In the figure,

- *cnf*(φ) is a set of clauses —possibly with newly introduced propositional variables— such that, for any assignment μ in the extended language:[2]
  - if μ satisfies *cnf*(φ) then the restriction of μ to the language of φ satisfies φ, and
  - if μ satisfies φ then there exists an assignment in the language of *cnf*(φ) which (i) extends μ and (ii) satisfies *cnf*(φ).
- *choose-literal*(φ, μ) returns a literal occurring in φ and chosen according to some heuristic criterion.
- if l is a literal, $\overline{l}$ stands for A if $l = \neg A$, and for A if $l = \neg A$;
- for any literal l and formula φ, *assign*(l, φ) is the formula obtained from φ by (i) deleting the clauses in which l occurs as a disjunct, and (ii) eliminating $\overline{l}$ from the others.

As can be observed, the procedure LSAT$_{\mathrm{DP}}$ in Figure 1 is the DP-procedure modulo the call to LCONSIST(μ) when it finds an assignment μ satisfying the input formula.

3.2 Lconsist(μ)

Whether an assignment is consistent, depends on the particular logic L being considered. Furthermore, depending on the logic L considered, the consistency problem for L (i.e., determining whether a formula is consistent in L) belongs to different complexity classes. In particular, the consistency problem for E, EM, EN, EMN is NP-complete, while for EC, ECN, EMC, EMCN it is PSPACE-complete (see [12, 9]). Here, to save space, we divide these eight logics in two groups. We present the algorithms for checking the L-consistency of an assignment first in the case in which L is one of E, EM, EN, EMN, and then in the case in which L is one of the others.

Logics E, EM, EN, EMN The following conditions are an easy consequence of the results presented in [12]. Let $\mu = \bigwedge_i \Box\alpha_i \wedge \bigwedge_j \neg\Box\beta_j \wedge \gamma$ be an assignment in which γ is a propositional formula. Let L be one of the logics E, EM, EN, EMN. μ is consistent in L if for each conjunct $\neg\Box\beta_j$ in μ one of the following conditions is satisfied:

- $(\alpha_i \equiv \neg\beta_j)$ is L-consistent for each conjunct $\Box\alpha_i$ in μ, and L=E;
- $(\alpha_i \wedge \neg\beta_j)$ is L-consistent for each conjunct $\Box\alpha_i$ in μ, and L=EM;
- $\neg\beta_j$ and $(\alpha_i \equiv \neg\beta_j)$ are L-consistent for each conjunct $\Box\alpha_i$ in μ, and L=EN;
- $\neg\beta_j$ and $(\alpha_i \wedge \neg\beta_j)$ are L-consistent for each conjunct $\Box\alpha_i$ in μ, and L=EMN.

When implementing the above conditions, care must be taken in order to avoid repetitions of consistency checks. In fact, while an exponential number of assignments satisfying the input formula can be generated by LSAT, at most n^2 checks are possible in L, where n is the number of "$\Box$" in the input formula. Given this upper bound, for each new consistency check, we can cache the result for a future possible re-utilization in a $n \times n$ matrix M. This ensures that at most n^2 consistency checks will be performed. In more detail, given an enumeration $\varphi_1, \varphi_2, \ldots, \varphi_n$ of the boxed subformulas of the input formula, M[i,j], with $i \neq j$, stores the result of the consistency check for $(\varphi_i \wedge \neg\varphi_j)$. M[i,i] stores the result of the consistency check for $\neg\varphi_i$. Initially, each element of the matrix M has value *Undef* (meaning that the corresponding test has not been done yet). The result is the procedure LCONSIST in Figure 2.

Consider Figure 2 and assume that L=E or L=EN. Given a pair of conjuncts $\Box\alpha_i$ and $\neg\Box\beta_j$, we split the consistency test for $(\alpha_i \equiv \neg\beta_j)$ in two simpler sub-tests:

[2] Let μ be an assignment in a language L. Let $L' \subseteq L$ be a language. The *restriction of* μ *to* L' is the assignment obtained from μ by deleting the conjuncts not in L'. Let μ' an assignment. μ' *extends* μ if each conjunct of μ is also a conjunct of μ'.

```
function LCONSIST(⋀_i □α_i ∧ ⋀_j ¬□β_j ∧ γ)
  foreach conjunct □β_j do
    foreach conjunct □α_i do
      if M[i,j] = Undef then M[i,j] := LSAT(α_i ∧ ¬β_j);
      if L ∈ {EN,EMN} and M[i,j] = True then M[j,j] := True;
      if L ∈ {E,EN} and M[i,j] = False then
        if M[j,i] = Undef then M[j,i] := LSAT(¬α_i ∧ β_j);
        if L = EN and M[j,i] = True then M[i,i] := True;
        if M[j,i] = False then return False
    end
    if L ∈ {EN,EMN} then
      if M[j,j] = Undef then M[j,j] := LSAT(¬β_j);
      if M[j,j] = False then return False
  end;
  return True.
```

Fig. 2. LCONSIST for E, EM, EN, EMN

- first, we test whether $(\alpha_i \wedge \neg\beta_j)$ is consistent, and
- only if this test gives *False*, we test whether $(\neg\alpha_i \wedge \beta_j)$ is consistent.

Notice also that, in case L=EN or L=EMN, if we know that, e.g., $(\alpha_i \wedge \neg\beta_j)$ is consistent, then also $\neg\beta_j$ is consistent and we store this result in M[j,j].

Logics EC, ECN, EMC, EMCN The following conditions are an easy consequence of the results presented in [12]. Let $\mu = \bigwedge_i \Box\alpha_i \ \wedge \ \bigwedge_j \neg\Box\beta_j \ \wedge \ \gamma$ be an assignment in which γ is a propositional formula. Let Δ be the set of formulas α_i such that $\Box\alpha_i$ is a conjunct of μ. Let L be one of logics EC, ECN, EMC, EMCN. μ is consistent in L if for each conjunct $\neg\Box\beta_j$ in μ one of the following conditions is satisfied:

- $((\bigwedge_{\alpha_i\in\Delta'} \alpha_i) \equiv \neg\beta_j)$ is L-consistent for each non empty subset Δ' of Δ, and L=EC;
- $((\bigwedge_{\alpha_i\in\Delta'} \alpha_i) \equiv \neg\beta_j)$ is L-consistent for each subset Δ' of Δ, and L=ECN;
- Δ is empty or $((\bigwedge_{\alpha_i\in\Delta} \alpha_i) \wedge \neg\beta_j)$ is L-consistent, and L=EMC;
- $((\bigwedge_{\alpha_i\in\Delta} \alpha_i) \wedge \neg\beta_j)$ is L-consistent, and L=EMCN.

Assume that L=EC or L=ECN. The straightforward implementation of the corresponding condition may lead to an exponential number of checks in the cardinality $|\Delta|$ of Δ. More carefully, for each conjunct $\neg\Box\beta_j$ in μ, we can perform at most $|\Delta| + 1$ checks if

1. for each formula α_i in Δ, we first check whether $(\neg\alpha_i \wedge \beta_j)$ is consistent in L. Let Δ' be the set of formulas for which the above test fails. Then,
2. in case L=ECN or $\Delta' \neq \emptyset$, we perform the last test, checking whether $((\bigwedge_{\alpha_i\in\Delta'} \alpha_i) \wedge \neg\beta_j)$ is consistent in L.

Furthermore, the result of the consistency checks performed in the first step can be cached in a matrix M analogous to the one used in the previous subsection.

4 Implementation and comparative analysis

*SAT is built on top of SATO ver. 3.2 [4]. We have chosen SATO as the basis for our system since it is fast and has many options, including various splitting heuristics and backjumping.

```
function LCONSIST(⋀_i □α_i ∧ ⋀_j ¬□β_j ∧ γ)
  Δ := {α_i | □α_i is a conjunct of μ};
  foreach conjunct □β_j do
    Δ' := Δ;
    if L ∈ {EC, ECN} then
      foreach conjunct □α_i do
        if M[j,i] = Undef then M[j,i] := LSAT(¬α_i ∧ β_j);
        if M[j,i] = True then Δ' = Δ' \ {α_i}
      end;
    if L ∈ {ECN, EMCN} or Δ' ≠ ∅ then
      if not LSAT(⋀_{α_i ∈ Δ'} α_i ∧ ¬β_j) then return False
  end;
  return True.
```

Fig. 3. LCONSIST for EC, ECN, EMC (R), EMCN (K)

We have inherited some of these options, and they are available for experimentation. Also, SATO has been written using some Software Engineering conventions which have made and will make much easier to tune it for our goals. Besides the options inherited from SATO, our system allows for other possibilities that we have developed while implementing the system. It is out of the goals of this paper to describe *SAT structure. See [13] and the manual distributed with *SAT for a more detailed presentation of the system. To understand the experimental results, it suffices to say that the core of *SAT is a C implementation of the procedures LSAT and LCONSIST in Figures 1, 2, 3. Also, we run *SAT with the following options enabled:

Early-pruning. Before each splitting step in LSAT, the L-consistency of the assignment generated so far is checked by a call to LCONSIST.

Caching for K. In the case of the logic K, *SAT uses an additional data structure which allows to associate to any formula φ the result of LSAT(φ). Before invoking LSAT on a formula ψ, LCONSIST checks whether the K-consistency of ψ has already been determined.

The availability of decision procedures for the logics we consider varies significantly. For EMCN, that we recall is equivalent to K, there are many implemented decision procedures available, see, e.g., [14, 15]. For E, EM and EMC, Gasquet and Herzig [16] provide a reduction to normal modal logics: by implementing this reduction we indirectly obtain decision procedures for these logics. Fitting [17] calls U the logic EM, and defines a tableau system for it. More recently, Governatori and Luppi [18] define a tableau-like proof system for classical, monotonic and regular modal logics. We are not aware of any implementation of this tableau system. For EN, EC, EMN and ECN we are not aware of any other implemented decision procedure, nor of any reduction into a formalism for which a decision procedure is available.

We restrict our analysis to E. In fact, both our decision procedures for E and EM, and Gasquet and Herzig's reductions for E and EM, are similar. We expect that the experimental analysis for EM would lead to results similar to the ones we have for E. For EMC, Gasquet and Herzig's reduction is to a normal modal logic for which we do not have a system available.[3]

Gasquet and Herzig [16] provide a translation which maps any formula φ into a formula φ_{GH} in K_2, i.e., the smallest normal modal logic with two modal operators $\Box_1$ and $\Box_2$. The translation

[3] The reduction maps the consistency problem for a formula φ in EMC, into the consistency problem for a formula φ' in the smallest normal modal logic with two modal operators $\Box_1$, $\Box_2$ and augmented with the schema $\psi \supset \Box_1\psi$.

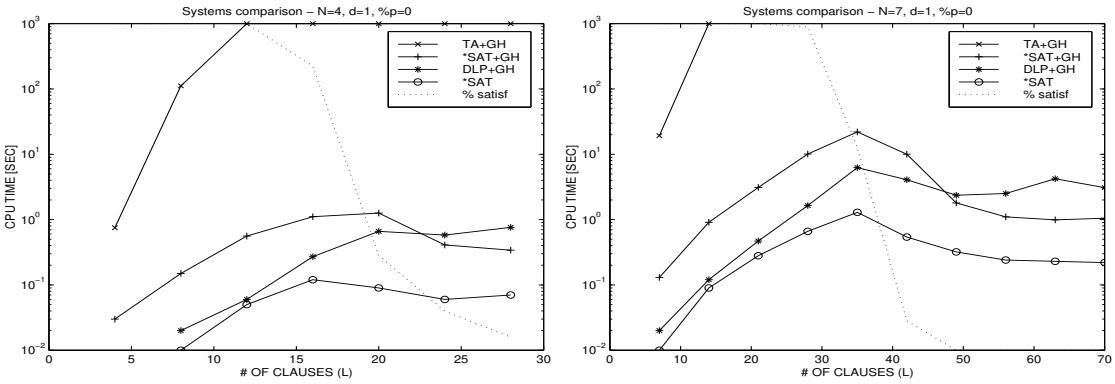

Fig. 4. Logic E. *SAT, *SAT+GH, DLP+GH, and TA+GH median CPU time. $N = 4, 7$. $p = 0\%$. 100 samples/point. Background: satisfiability percentage.

is such that φ is satisfiable in E iff φ_{GH} is satisfiable in K_2. This translation is defined in the following way:

- $\varphi_{GH} = \varphi$, if φ is a propositional variable,
- $\varphi_{GH} = \neg\Box_1\neg(\Box_2\psi_{GH} \wedge \Box_1\neg\psi_{GH})$, if $\varphi = \Box\psi$,

and homomorphic for the cases of the propositional connectives.

Consider a formula φ. We compare *SAT performances on φ with respect to *SAT, DLP and TA performances on φ_{GH}. To make evident when a system is run using Gasquet and Herzig's translation, we append the string "+GH" to the name of the system. Therefore, in the following, we will have the systems *SAT, *SAT+GH, DLP+GH, and TA+GH.

In E, the $3\mathrm{CNF}_K$ test methodology is not suited. Indeed, it is no longer the case that for any modal formula φ there exists a $3\mathrm{CNF}_K$ formula which is E-satisfiable iff φ is E-satisfiable. Furthermore, checking the consistency of an assignment μ in E amounts to determine the consistency of $(\alpha \equiv \neg\beta)$ for each pair of conjuncts $\Box\alpha$ and $\neg\Box\beta$ in μ: most of these tests, in case α and β are $3\mathrm{CNF}_K$ clauses, can be trivially satisfied.

We therefore consider sets of $3\mathrm{CNF}_E$ formulas. A $3\mathrm{CNF}_E$ formula is a conjunction of $3\mathrm{CNF}_E$ clauses, each with three disjuncts. Each disjunct in a $3\mathrm{CNF}_E$ clause is either a propositional literal or a formula having the form $\Box C$ or $\neg\Box C$, where C is a $3\mathrm{CNF}_E$ formula. For example,

$$\Box((\Box(l_1 \vee l_2 \vee l_3) \vee l_4 \vee l_5) \wedge (l_6 \vee l_7 \vee l_8))$$

where each l_i $(1 \leq i \leq 8)$ is a propositional literal, is a $3\mathrm{CNF}_E$ formula. For any formula φ, there exist a $3\mathrm{CNF}_E$ formula which is E-consistent iff φ is E-consistent.

Sets of $3\mathrm{CNF}_E$ formulas can be randomly generated according to the following parameters:

(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 a disjunct occurring in a clause at depth $< d$ is purely propositional.
(v) the number C of clauses at depth $d > 0$

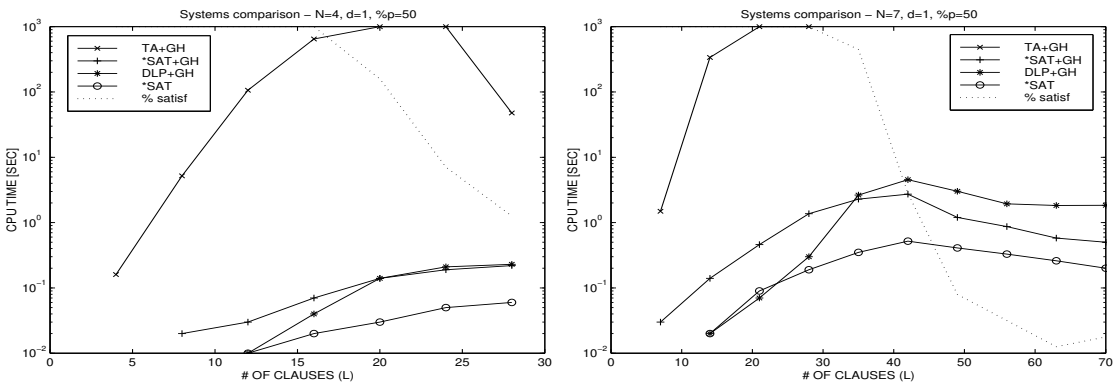

Fig. 5. Logic E. *SAT, *SAT+GH, DLP+GH, and TA+GH median CPU time. $N = 4, 7$. $p = 50\%$. 100 samples/point. Background: satisfiability percentage.

Parameters (i) through (iv) are the same used used to generate 3CNF_K formulas. A 3CNF_K is thus a 3CNF_E formula in which $C = 1$.

A problem set is characterized by N and p: d and C are fixed to 1 and L respectively; L is given increasing values in such a way to empirically cover the "100% satisfiable – 100% unsatisfiable" transition. We also check that in each sample there are no multiple occurrences of a formula in a clause, at the same ensuring that the propositional vs. the modal structure of the formula only depends on p. Notice that while increasing L also C is increased. As a consequence, for each pair of formulas $\Box\alpha_i$ and $\neg\Box\beta_j$ in an assignment satisfying a 3CNF_E formula, the recursive E-consistency check for $(\alpha_i \equiv \neg\beta_j)$ has itself a phase transition from 100% satisfiable to 100% unsatisfiable when increasing L. Overall, for low [resp. high] values of L we expect that each satisfying assignment should be trivially determined to be E-consistent [resp. not E-consistent].

For each L in a problem set, 100 3CNF_E formulas are randomly generated, and the resulting formulas are given in input to the procedure under test. A timeout stops the execution of the system on a formula after 1000 seconds of CPU time. We consider the following problems sets:

- PEN4p0, PEN5p0, PEN6p0, PEN7p0 in which $p = 0\%$ while $L = 4, 5, 6, 7$ respectively, and
- PEN4p50, PEN5p50, PEN6p50, PEN7p50 in which $p = 50\%$ while $L = 4, 5, 6, 7$ respectively.

Given the huge amount of time that FLOTTER takes to prepare the formula for SPASS, we run TA+GH only on the problems sets PEN4p0 and PEN4p50. For PEN7p0 and PEN7p50, we run TA+GH only on the initial points. We only take into account the time the systems take for the main processing of the formula. In particular, for each system, we do not take into account the time needed to perform the Gasquet and Herzig's conversion; and for TA+GH we take into account only the time taken by SPASS. For the lack of space, only the median plots of the systems on PEN4p0 and PEN7p0 are shown in Figure 4. The behavior of *SAT, *SAT+GH and DLP+GH on PEN5p0 and PEN6p0 is indeed very similar to what is shown in Figure 4.

Consider Figure 4. As can be observed, *SAT is the fastest: the gap with the other systems is of more than one order of magnitude for certain values of L. However, both *SAT+GH and DLP+GH perform quite well, better than one could have imagined given that the consistency problem for E and K_2 belongs to two different complexity classes. However, a closer look to Gasquet and Herzig's reduction reveals that, considering a 3CNF_E formula φ, and an assignment $\mu = \bigwedge_{i=1}^{m} \Box\alpha_i \wedge \bigwedge_{j=1}^{n} \neg\Box\beta_j \wedge \gamma$ (as usual we assume that γ is a propositional formula) in the language of φ,

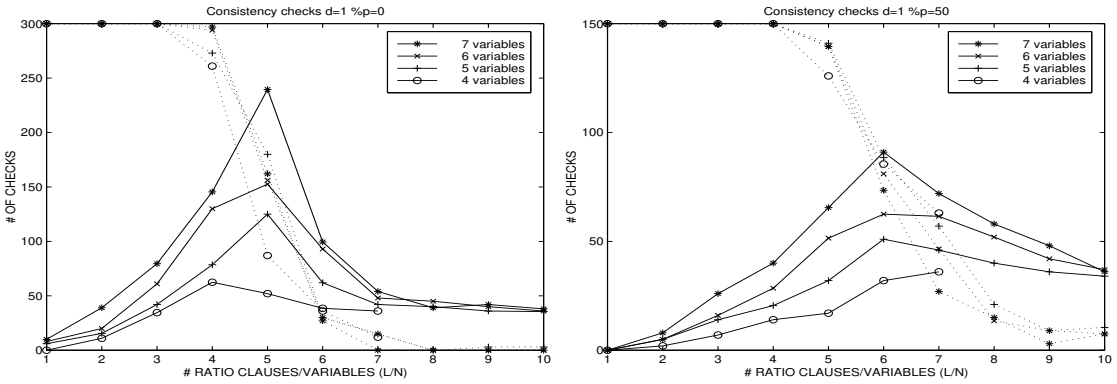

Fig. 6. Logic E. *SAT median number of calls to LSAT. $N = 4, 5, 6, 7$. $p = 0\%$ (left) and $p = 50\%$ (right). 100 samples/point. Background: satisfiability percentages.

1. μ satisfies φ iff μ_{GH} satisfies φ_{GH}.
2. for checking the E-consistency of μ, *SAT performs at most $2mn$ consistency checks involving the formulas $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m$.
3. for checking the K_2-consistency of μ_{GH}, both *SAT+GH and DLP+GH perform at most $2mn$ consistency checks involving the formulas $\alpha_{1GH}, \ldots, \alpha_{mGH}, \beta_{1GH}, \ldots, \beta_{mGH}$.

The first two points are obvious. To understand the last, it suffices to notice that μ_{GH} is propositionally equivalent to

$$\bigwedge_{i=1}^{m} \neg\Box_1\neg(\Box_2\alpha_{iGH} \wedge \Box_1\neg\alpha_{iGH}) \ \wedge \ \bigwedge_{j=1}^{n} \Box_1(\neg\Box_2\beta_{jGH} \vee \neg\Box_1\neg\beta_{jGH}) \ \wedge \ \gamma.$$

Given that both *SAT+GH and DLP+GH use caching, these procedures will perform at most a quadratic number of checks in the number of subformulas of φ_{GH}. This is not the case for TA+GH, since SPASS does not have any caching mechanism. This explains the good behavior of *SAT+GH and DLP+GH, and the bad behavior of TA+GH.

For $N = 4, 7$ and $p = 50\%$, *SAT, *SAT+GH, DLP+GH and TA+GH median times are plotted in Figure 5. As before, for the lack of space, we did not show the results for PEN5p50 and PEN6p50. As it can be observed, the situation is very similar to the case in which $p = 0\%$. The only difference is that now *SAT+GH performs better than DLP+GH for a lower value of L. This is reasonable, since for each L, the number of consistency checks performed by *SAT+GH because of early pruning, diminishes when p increases.

Finally, notice the easy-hard-easy pattern of *SAT. To better appreciate it, Figure 6 shows the number of calls to LSAT done by LCONSIST on PEN4p0-PEN7p0 (left), PEN4p50-PEN7p50 (right) against the ratio $\frac{L}{N}$ between L and N. As can be observed, *SAT performs a number of LSAT calls whose maximum roughly correspond to the 50% of satisfiable formulas. This transition happens when $\frac{L}{N}$ is close to 5 for $p = 0\%$ and to 6 for $p = 50\%$. This behavior reflects the above stated intuition according to which for low [resp. high] values of L all the formulas should be easily determined to be E-consistent [resp. not E-consistent]. Notice also that:

- the easy-hard-easy pattern in the plots of Figure 6 becomes more evident when increasing the number of variables and the phenomenon is even clearer when $p = 0\%$; and
- the shape of the transition seems to be independent from the number of variables.

5 Conclusions

We have presented a set of SAT-based decision procedures for eight classical modal logics. We implemented the decision procedures in our system *SAT which is the only system that is able to deal with EN, EC, ECN and EMN. We have defined a testing methodology which generalizes the $3CNF_K$ methodology by Giunchiglia and Sebastiani [2], and which is suitable for testing systems for non-normal modal logics. The experimental evaluation shows that *SAT+GH performs better than or as well as other state of the art systems on sets of randomly generated $3CNF_E$ formulas. In all the tests that we considered, *SAT outperforms systems exploiting Gasquet and Herzig's reduction to multimodal normal logics.

References

1. E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Proc. KR'98*, Trento, Italy, June 1998.
2. 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.
3. F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for ALC. In *Proc. of KR'96*, Cambridge, MA, USA, November 1996.
4. 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.
5. E. Giunchiglia, F. Giunchiglia, and A. Tacchella. *SAT, KSATC, DLP and TA: a comparative analysis. In *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.
6. B. F. Chellas. *Modal Logic – an Introduction*. Cambridge University Press, 1980.
7. Richard Montague. Pragmatics. In R. Klibansky, editor, *Contemporary Philosophy: A Survey. I*, pages 102–122. La Nuova Italia Editrice, Florence, 1968.
8. Krister Segerberg. *An Essay in Classical Modal Logic*. Philosophical Studies, Uppsala, 1 edition, 1971.
9. R. Fagin, J.Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*. MIT Press, 1995.
10. R. Sebastiani and F. Giunchiglia. From Tableau-based to SAT-based procedures - preliminary report. Technical Report 9711-14, IRST, Trento, Italy, November 1997.
11. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
12. Moshe Y. Vardi. On the complexity of epistemic reasoning. In *Proceedings, Fourth Annual Symposium on Logic in Computer Science*, pages 243–252, Asilomar Conference Center, Pacific Grove, California, 5–8 June 1989. IEEE Computer Society Press.
13. Armando Tacchella. *SAT system description. In *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.
14. 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)*. CEUR, May 1998.
15. 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.
16. O. Gasquet and A. Herzig. From classical to normal modal logics. In Heinrich Wansing, editor, *Proof Theory of Modal Logics*, volume 2 of *Applied Logic Series*, pages 293–311. Kluwer Academic Publishers, 1996.
17. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing, 1983.
18. G. Governatori and A. Luppi. Labelled tableaux for non-normal modal logics. In *Sixth Conference of the Italian Association for Artificial Intelligence, AI*IA '99, Bologna, Italy, September 14–17, 1999: proceedings*, 1999.