Theoretical Computer Science 606 (2015) 25–41

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Symmetric blocking ☆

Carlos Areces [a,b], Ezequiel Orbe [a,b,*]

[a] FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
[b] CONICET, Argentina

ARTICLE INFO

Article history:
Received 6 September 2014
Received in revised form 19 May 2015
Accepted 5 June 2015
Available online 10 June 2015

Keywords:
Modal logics
Symmetry
Blocking
Detection
Evaluation

ABSTRACT

We present three different techniques that use information about symmetries detected in the input formula to block the expansion of diamonds in a modal tableau. We show how these blocking techniques can be included in a standard tableaux calculus for the basic modal logic, and prove that they preserve soundness and completeness. We empirically evaluate these blocking mechanisms in different modal benchmarks.

1. Introduction

In the context of automated reasoning a symmetry can be defined as a permutation of the variables (or literals) of a problem that preserves its structure and its set of solutions. Symmetries have been extensively investigated and successfully exploited for propositional satisfiability (SAT). Already in [1], Krishnamurthy introduced symmetry inference rules to strengthen resolution-based proof systems for propositional logic leading to much shorter proofs of certain difficult problems (e.g., the pigeonhole problem). Since then, many articles discuss how to detect and exploit symmetries for propositional logic. Most of them can be grouped into two different approaches: static symmetry breaking (e.g., [2–4]) and dynamic symmetry breaking (e.g., [5,6]). In the former, symmetries are detected and eliminated from the problem statement before a SAT solver is used, i.e., they work as a preprocessing step. In the latter, symmetries are detected and broken during the search space exploration. Independently of the particular characteristics of each approach, they share the same goal: the goal is to identify symmetric branches of the search space and guide the SAT solver away from symmetric branches already explored.

Similar techniques for logics other than propositional logic have been investigated in the last years, e.g. [7,8]. To the best of our knowledge, symmetries remain largely unexplored in automated theorem proving for modal logics. In [9], we have laid the theoretical foundations to exploit symmetries in a number of modal logics, and developed techniques to efficiently detect symmetries in an input formula. In this paper we put these techniques to work and show how to use symmetry information in a modal tableaux calculus by presenting novel blocking mechanisms. These techniques, that we called *symmetric blocking*, dynamically delay the application of the rule that expands a yet unexplored diamond formula, if the rule has been already applied to a symmetric formula. In the last part of the article, we carry out an empirical evaluation of the effectiveness of these blocking mechanisms.

☆ This work was partially supported by grants ANPCyT-PICT-2013-2011, ANPCyT-PICT-2010-688, the FP7-PEOPLE-2011-IRSES Project "Mobility between Europe and Argentina applying Logics to Systems" (MEALS) and the Laboratoire International Associé "INFINIS".

* Corresponding author at: FaMAF, Haya de la Torre S/N, Córdoba, Argentina. CP:5000.
E-mail addresses: carlos.areces@gmail.com (C. Areces), orbe@famaf.unc.edu.ar (E. Orbe).

http://dx.doi.org/10.1016/j.tcs.2015.06.020
0304-3975/

Outline. Section 2 introduces the required definitions on modal language and symmetries. Section 3 briefly discusses the algorithm used to detect symmetries in modal formulas and shows experimental data about the detection construction. Section 4 presents a classic labeled tableaux calculus for the basic modal logic and introduces different symmetric blocking conditions. We also show that the resulting calculi are terminating, sound and complete. Finally, Section 5 presents experimental results about the effects of symmetric blocking in modal benchmarks. Section 6 concludes with some final remarks.

2. Basic definitions

In this section we introduce the basic notions concerning modal languages, permutations and symmetries. In what follows, we will assume basic knowledge of classical modal logics and refer the reader to [10,11] for technical details.

We will discuss only the mono-modal basic modal logic. The results we establish extend, in an obvious way, to the multi-modal case. In Section 6 we discuss other modal logics.

Definition 1 *(Syntax).* Let $\mathsf{PROP} = \{p_1, p_2, \ldots\}$ be a countably infinite set of *propositional variables*. The well-formed formulas of the basic modal logic are defined by the rule

$$\varphi, \psi := p \mid \neg p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg\Box\varphi \mid \Box\varphi,$$

where $p \in \mathsf{PROP}$. FORM denotes the set of all well-formed formulas of the basic modal logic. Given a formula $\varphi \in \mathsf{FORM}$, define $\mathsf{PROP}(\varphi)$ as the set of propositional variables occurring in φ; for S a set of formulas, let $\mathsf{PROP}(S) = \bigcup_{\varphi\in S} \mathsf{PROP}(\varphi)$.

Definition 2 *(Propositional literals).* A *propositional literal* l is either a propositional variable $p \in \mathsf{PROP}$ or its negation $\neg p$. The set of propositional literals over PROP is $\mathsf{PLIT} = \mathsf{PROP} \cup \{\neg p \mid p \in \mathsf{PROP}\}$. A set of propositional literals L is *complete* if for each $p \in \mathsf{PROP}$ either $p \in L$ or $\neg p \in L$. It is *consistent* if for each $p \in \mathsf{PROP}$ either $p \notin L$ or $\neg p \notin L$. Any complete and consistent set of literals L defines a unique valuation $v \subseteq \mathsf{PROP}$ which is defined as $p \in v$ if and only if $p \in L$. For $S \subseteq \mathsf{PROP}$, the *consistent and complete set of literals generated by* S (notation: L_S) is $S \cup \{\neg p \mid p \in \mathsf{PROP} \setminus S\}$.

Definition 3 *(Modal literals, clauses and modal CNF).* A formula is in *modal conjunctive normal form (modal CNF)* if it is a conjunction of clauses. A *clause* is a disjunction of propositional and *modal literals*. A *modal literal* is a formula of the form $\Box C$ or $\neg\Box C$ where C is a clause.

The function clauses returns the multiset of clauses in a formula φ. Let $\uplus$ be the operation of union with repetition between multisets, we define clauses as follows

$$\begin{array}{lcl}
\mathsf{clauses}(p) & = & \{\} \\
\mathsf{clauses}(\neg p) & = & \{\} \\
\mathsf{clauses}(\Box C) & = & \mathsf{clauses}(C) \\
\mathsf{clauses}(\neg\Box C) & = & \mathsf{clauses}(C) \\
\mathsf{clauses}(C) & = & \{C\} \uplus \biguplus_{l\in C} \mathsf{clauses}(l) \\
\mathsf{clauses}(\varphi) & = & \biguplus_{C\in\varphi} \mathsf{clauses}(C).
\end{array}$$

Example 1. The formula $\neg\Box(\neg p \vee \Box q \vee \Box\neg q \vee \neg p) \wedge \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$ is in modal CNF. It is the conjunction of two clauses $\neg\Box(\neg p \vee \Box q \vee \Box\neg q \vee \neg p)$ and $\neg\Box(\neg q \vee \Box p \vee \Box\neg p)$ which are also modal literals (notice that, in this case, each clause contains only one disjunct).

Every modal formula can be transformed into an equivalent formula in modal CNF at the risk of an exponential blowup in the size of the formula; it can be transformed into an equisatisfiable formula in polynomial time, using additional propositional variables (see [12] for details).

In what follows, we assume that modal formulas are in modal CNF. Moreover, we want to consider formulas modulo commutativity and idempotency of conjunction and disjunction. To that end, we represent a modal CNF formula as a *set* of clauses (interpreted conjunctively), and each clause as a *set* of propositional and modal literals (interpreted disjunctively). This set representation disregards order and multiplicity of clauses and literals in a formula. We will assume that modal formulas are represented using set notation, even though we will often write them using the familiar notation.

Example 2. The formula $\neg\Box(\neg p \vee \Box q \vee \Box\neg q \vee \neg p) \wedge \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$ is written, using the set notation, as $\{\neg\Box\{\neg p, \Box\{q\}, \Box\{\neg q\}\}, \neg\Box\{\neg q, \Box\{p\}, \Box\{\neg p\}\}\}$.

Definition 4 *(Semantics).* A *pointed model* is a tuple $\langle w, W, R, V\rangle$, where W is a non-empty set, $w \in W$, $R \subseteq W \times W$ and $V(v) \subseteq \mathsf{PROP}$ for all $v \in W$. Let $\mathcal{M} = \langle w, W, R, V\rangle$ be a pointed model and $w' \in W$, define $\mathcal{M}, w' = \langle w', W, R, V\rangle$.

Let $\mathcal{M} = \langle w, W, R, V\rangle$ be a pointed model, we define the satisfiability relation $\models$ for modal CNF formulas, clauses and literals as follows. Let φ be a modal CNF formula, C a modal clause, and $p \in \mathsf{PROP}$, then

$$\begin{array}{lll}
\mathcal{M} \models p & \text{iff} & p \in V(w) \\
\mathcal{M} \models \neg p & \text{iff} & p \notin V(w) \\
\mathcal{M} \models \Box C & \text{iff} & \langle w', W, R, V\rangle \models C, \text{ for all } w' \text{ such that } wRw' \\
\mathcal{M} \models \neg\Box C & \text{iff} & \mathcal{M} \not\models \Box C \\
\mathcal{M} \models C & \text{iff} & \text{there is some literal } l \in C \text{ such that } \mathcal{M} \models l \\
\mathcal{M} \models \varphi & \text{iff} & \text{for all clauses } C \in \varphi \text{ we have } \mathcal{M} \models C.
\end{array}$$

We say that a formula φ is *satisfiable* if for some pointed model $\mathcal{M}$ we have that $\mathcal{M} \models \varphi$. Otherwise, it is unsatisfiable.

Definition 5 *(Modal depth).* The modal depth of a modal CNF formula φ (notation: $\mathsf{md}(\varphi)$) is the maximum nesting of $\Box$ operators that occurs in φ. Let φ be a modal CNF formula, C a modal clause, and $p \in \mathsf{PROP}$, then

$$\begin{array}{lcl}
\mathsf{md}(p) & = & 0 \\
\mathsf{md}(\neg p) & = & 0 \\
\mathsf{md}(\Box C) & = & 1 + \mathsf{md}(C) \\
\mathsf{md}(\neg\Box C) & = & 1 + \mathsf{md}(C) \\
\mathsf{md}(C) & = & \max\{\mathsf{md}(l) \mid l \in C\} \\
\mathsf{md}(\varphi) & = & \max\{\mathsf{md}(C) \mid C \in \varphi\}.
\end{array}$$

Definition 6 *(Permutations over* PLIT*).* A *permutation* is a bijective function $\rho : \mathsf{PLIT} \mapsto \mathsf{PLIT}$. If L is a set of literals then $\rho(L) = \{\rho(l) \mid l \in L\}$.

For $p \in \mathsf{PROP}$, let $\sim\neg p = p$ and $\sim p = \neg p$. We say that a permutation ρ is *consistent* if for every literal l, $\rho(\sim l) = \sim\rho(l)$. We say that a permutation ρ is a *symmetry* of φ if $\varphi = \rho(\varphi)$ (here it is important that φ is represented using set notation).

We say that a permutation ρ *has finite support* if for only a finite number of literals we have that $\rho(l) \neq l$, i.e., ρ is the identity for most literals in PLIT. Permutations with finite support can be represented using *cyclic notation* (see [13] for details). A *cycle* is a finite sequence of literals, and a permutation is represented as a finite sequence of cycles such that any literal appears only once. Let $(l_{11}\ \ldots\ l_{1k_1})\ldots(l_{i1}\ \ldots\ l_{ik_i})$ be a finite sequence of cycles, it defines the permutation

$$\begin{array}{lll}
\rho(l) & = l & \text{if } l \text{ is a literal that does not appear in the sequence} \\
\rho(l_{ij}) & = l_{ij+1} & \text{if } j < k_i \\
\rho(l_{ik_i}) & = l_{i1}. &
\end{array}$$

Example 3. $\rho = (p\ \neg q)(\neg p\ q)$ is the permutation that makes $\rho(p) = \neg q$, $\rho(\neg q) = p$, $\rho(\neg p) = q$, $\rho(q) = \neg p$ and leaves unchanged all other literals; $\rho = (p\ q\ r)(\neg p\ \neg q\ \neg r)$ is the permutation $\rho(p) = q$, $\rho(q) = r$ and $\rho(r) = p$ and similarly for the negations.

Consider the formula $\varphi = (\neg p \vee r) \wedge (q \vee r) \wedge \Box(\neg p \vee q)$. The permutation $\rho = (p\ \neg q)(\neg p\ q)$ is a consistent permutation. Moreover, ρ is a symmetry of φ. Notice that φ and $\rho(\varphi)$ are identical when using set notation.

In modal logics that enjoy the tree model property there is a direct correlation between the modal depth of the formula and the depth in a tree model satisfying it: in models, a notion of layer is induced by the depth (distance from the root) of the nodes, whereas in formulas, a notion of layer is induced by the nesting of the modal operators. A consequence of this correspondence is that propositional literals at different modal depths are semantically independent of each other (see [14] for details), i.e., a propositional literal can be assigned a different value in each different layer.

Definition 7 *(Layered permutation).* A *layered permutation* $\bar{\rho}$ is a, possible empty, finite sequence of permutations $\langle\rho_1, \ldots, \rho_k\rangle$. Let $|\langle\rho_1, \ldots, \rho_k\rangle| = k$ be the length of $\bar{\rho}$ ($\langle\rangle$ has length 0). For $1 \leq i \leq n$, $\bar{\rho}_i$ is the sub-sequence that starts from the i^{th} element of $\bar{\rho}$ (in particular, $\bar{\rho}_i = \langle\rangle$ for $i > |\bar{\rho}|$ and $\bar{\rho}_1 = \bar{\rho}$). Let $\bar{\rho} = \langle\rho_1, \ldots, \rho_k\rangle$ then $\bar{\rho}(i)$ is ρ_i if $1 \leq i \leq |\bar{\rho}|$ and $\bar{\rho}(i) = \rho_{Id}$ otherwise, for ρ_{Id} the identity permutation.

A layered permutation is *consistent* if all its permutations are consistent.

Let φ be a modal CNF formula, C a clause and $p \in \mathsf{PROP}$, we define the application of a layered permutation $\bar{\rho}$ to a modal CNF formula, a clause or literal as follows. Define $\langle\rangle(\varphi) = \varphi$, and for $|\bar{\rho}| \geq 1$ define

$$\begin{array}{lcl}
\bar{\rho}(p) & = & \bar{\rho}(1)(p) \\
\bar{\rho}(\neg p) & = & \bar{\rho}(1)(\neg p) \\
\bar{\rho}(\Box C) & = & \Box\bar{\rho}_2(C) \\
\bar{\rho}(\neg\Box C) & = & \neg\Box\bar{\rho}_2(C) \\
\bar{\rho}(C) & = & \{\bar{\rho}(l) \mid l \in C\} \\
\bar{\rho}(\varphi) & = & \{\bar{\rho}(C) \mid C \in \varphi\}.
\end{array}$$

As before, a layered permutation $\bar{\rho}$ is a symmetry of φ if $\bar{\rho}(\varphi) = \varphi$, when φ is represented using set notation.

In what follows we assume that all permutations are consistent.

Example 4. Layered permutations capture symmetries that non-layered permutations cannot define. Consider the formula $\varphi = (p \vee \Box(p \vee \neg r)) \wedge (\neg q \vee \Box(\neg p \vee r))$. If we only consider non-layered permutations then φ has no symmetry. However, the layered permutation $\langle \rho_1, \rho_2 \rangle$ generated by $\rho_1 = (p\ \neg q)(\neg p\ q)$ and $\rho_2 = (p\ \neg r)(\neg p\ r)$ is a symmetry of φ.

To make the article self-contained, we introduce here the needed definitions and results from [9].

Definition 8 *($\bar{\rho}$-simulation).* Let $\bar{\rho}$ be a layered permutation. A *$\bar{\rho}$-simulation* between two pointed models $\mathcal{M} = \langle w, W, R, V \rangle$ and $\mathcal{M}' = \langle w', W', R', V' \rangle$ is a (possibly infinite) family of relations $\{Z_{\bar{\rho}_i} \mid 1 \leq i\}$, with $Z_{\bar{\rho}_i} \subseteq W \times W'$ that satisfies the following conditions:

Root: $w Z_{\bar{\rho}_1} w'$.
$\bar{\rho}$-Harmony: $w Z_{\bar{\rho}_i} w'$ implies $l \in L_{V(w)}$ iff $\bar{\rho}_i(1)(l) \in L_{V'(w')}$.
Zig: If $w Z_{\bar{\rho}_i} w'$ and wRv then $v Z_{\bar{\rho}_{i+1}} v'$ for some v' such that $w'R'v'$.
Zag: If $w Z_{\bar{\rho}_i} w'$ and $w'R'v'$ then $v Z_{\bar{\rho}_{i+1}} v'$ for some v such that wRv.

We say that two pointed models $\mathcal{M}$ and $\mathcal{M}'$ are $\bar{\rho}$-similar (notation: $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}'$), if there is a $\bar{\rho}$-simulation between them.

Notice that the relation $\underline{\rightarrow}_{\bar{\rho}}$ is not symmetric (hence, it is not a bisimulation): $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}'$ does not imply $\mathcal{M}' \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}$. On the other hand, let $\bar{\rho} = \langle \rho_1, \ldots, \rho_k \rangle$ and $\bar{\rho}^{-1} = \langle \rho_1^{-1}, \ldots, \rho_k^{-1} \rangle$, then it is not difficult to prove that $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}'$ implies $\mathcal{M}' \underline{\rightarrow}_{\bar{\rho}^{-1}} \mathcal{M}$.

From the definition of $\bar{\rho}$-simulations it follows that while they do not preserve validity of modal formulas (as it is the case with bisimulations) they do preserve validity of *permutations* of formulas.

Proposition 1. *Let $\bar{\rho}$ be a consistent layered permutation, φ a modal CNF formula and $\mathcal{M} = \langle w, W, R, V \rangle$, $\mathcal{M}' = \langle w', W', R', V' \rangle$ models such that $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}'$. Then $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \bar{\rho}(\varphi)$.*

Given a tree model and a layered permutation, we can construct a new model as follows.

Definition 9 *($\bar{\rho}$-image of a tree model).* Given a pointed tree model $\mathcal{M} = \langle w, W, R, V \rangle$ and a layered permutation $\bar{\rho}$. Let $\mathsf{depth}(v)$ be the distance of v from the root w (in particular $\mathsf{depth}(w) = 0$). Define the *$\bar{\rho}$-image of $\mathcal{M}$* as the model $\mathcal{M}_{\bar{\rho}} = \langle w, W, R, V_{\bar{\rho}} \rangle$ where

$$V_{\bar{\rho}}(v) = \bar{\rho}(\mathsf{depth}(v) + 1)(L_{V(v)}) \cap \mathsf{PROP}.$$

It follows that $\mathcal{M}$ and $\mathcal{M}_{\bar{\rho}}$ are $\bar{\rho}$-similar.

Proposition 2. *Let $\bar{\rho}$ be a consistent layered permutation and $\mathcal{M}$ a pointed tree model. Then $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}_{\bar{\rho}}$.*

If φ is true in a model $\mathcal{M}$, $\bar{\rho}(\varphi)$ is true in $\mathcal{M}_{\bar{\rho}}$.

Proposition 3. *Let $\bar{\rho}$ be a consistent layered permutation, φ a modal CNF formula and $\mathcal{M}$ a pointed tree model. Then $\mathcal{M} \models \varphi$ if and only if $\mathcal{M}_{\bar{\rho}} \models \bar{\rho}(\varphi)$.*

3. Detecting modal symmetries

In [9], we presented a graph-based technique for the detection of symmetries in modal CNF formulas. In propositional logic, a standard approach is to reduce the problem of finding symmetries in formulas to the problem of finding automorphisms in labeled graphs: a labeled graph is constructed from a given formula in such a way that its automorphism group is isomorphic to the symmetry group of the formula [4,2,3].

The graph automorphism problem is known to be in the NP complexity class. It is not known to be in P, and it has not been proved NP-complete either [15]. Nevertheless there exist polynomial time algorithms for many special cases [16], and there are many efficient tools [17,18] capable of efficiently computing a set of generators [13] for the automorphism group of very large graphs.

In [9] we extended the graph construction for propositional formulas presented in [4] to a wide range of modal logics. For completeness, we briefly describe the graph construction that can be used to detect layered symmetries for the basic modal logic.

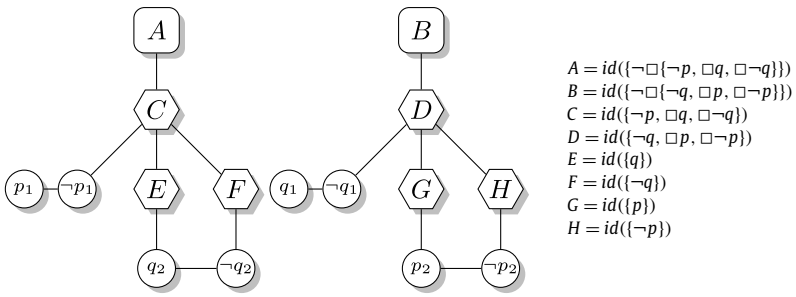


Fig. 1. Labeled graph associated to $\varphi = \neg\Box(\neg p \vee \Box q \vee \Box\neg q) \wedge \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$.

Definition 10. An undirected, labeled finite graph is a graph $G = (N, E, l)$ where N, the set of *nodes*, is a finite non-empty set; E, the set of *edges*, is a symmetric relation over $N \times N$; and $l : N \mapsto L$ is the *labeling function* mapping nodes into a finite set L of labels.

Definition 11 *(Layered graph construction).* Let φ be a modal CNF formula. Let $(\_)^{\mathrm{id}} : \mathsf{clauses}(\varphi) \mapsto I$ be an injective function that assigns each clause in φ a unique identifier over an arbitrary set I. Let $\mathsf{PROP}(\varphi, n)$ be the set of propositional symbols occurring in φ at modal depth n.

The graph $LG^{id}(\varphi) = (N, E, l)$ corresponding to φ and id is the smallest labeled graph satisfying the following conditions:

1. For each propositional variable $p \in \mathsf{PROP}(\varphi, i)$ with $0 \leq i \leq \mathsf{md}(\varphi)$:
   (a) There are nodes (p, i) and $(\neg p, i)$ in N and $l((p, i)) = l((\neg p, i)) = 0$.
   (b) There is an edge between (p, i) and $(\neg p, i)$ in E.
2. For each clause C at modal depth 0 there is a node C^{id} in N and $l(C^{id}) = 1$.
3. For each propositional literal l at modal depth i occurring as a disjunct in a clause C, there is an edge between C^{id} and (l, i) in E.
4. For each modal literal $\Box D$ occurring as a disjunct in a clause C:
   (a) There is a node D^{id} in N and $l(D^{id}) = 2$
   (b) There is an edge between D^{id} and C^{id} in E.
5. For each modal literal $\neg\Box D$ occurring as a disjunct in a clause C:
   (a) There is a node D^{id} in N and $l(D^{id}) = 3$
   (b) There is an edge between D^{id} and C^{id} in E.

Let n be the number of propositional variables appearing in φ then, $LG^{id}(\varphi)$ has at most $2n(\mathsf{md}(\varphi) + 1) + |\mathsf{clauses}(\varphi)|$ nodes.

Notice the way literals are handled during the construction of $LG^{id}(\varphi)$: the construction duplicates literal nodes occurring at different modal depth. By doing this we incorporate the notion of layering introduced in Section 2. Also, modal literals $\Box C$ and $\neg\Box C$ are labeled differently. This avoids spurious permutations mapping $\Box C$ literals to $\neg\Box C$ literals.

Example 5. Consider the following modal CNF formula $\varphi = \neg\Box(\neg p \vee \Box q \vee \Box\neg q) \wedge \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$. Using set representation φ is

$$\varphi = \{\, \{\neg\Box\{\neg p, \Box\{q\}, \Box\{\neg q\}\}\}, \{\neg\Box\{\neg q, \Box\{p\}, \Box\{\neg p\}\}\} \,\}$$

with eight clauses (2 at modal depth 0, 2 at modal depth 1, and 4 at modal depth 2) and six literals (2 at modal depth 1, and 4 at modal depth 2).

Fig. 1 shows its associated labeled graph $LG^{id}(\varphi)$ (labels are represented by shapes, and nodes (l, i) by l_i in the figure).

A set of generators for the automorphisms group of the graph is given by:

$$\pi_1 = (G\ H)(p_2\ \neg p_2),$$
$$\pi_2 = (E\ F)(q_2\ \neg q_2),$$
$$\pi_3 = (A\ B)(C\ D)(E\ G)(F\ H)(p_1\ q_1)(\neg p_1\ \neg q_1)(p_2\ q_2)(\neg p_2\ \neg q_2).$$

To obtain the corresponding generators for the formula φ, we restrict the automorphisms of the graph to the propositional literals nodes. The index associated to each propositional literal node indicates its modal depth and, therefore, given a cycle permuting such nodes, we can determine to which permutation in the layered permutation it corresponds.

We obtain the following generators for the symmetry group of φ (ρ_{Id} is the identity permutation):

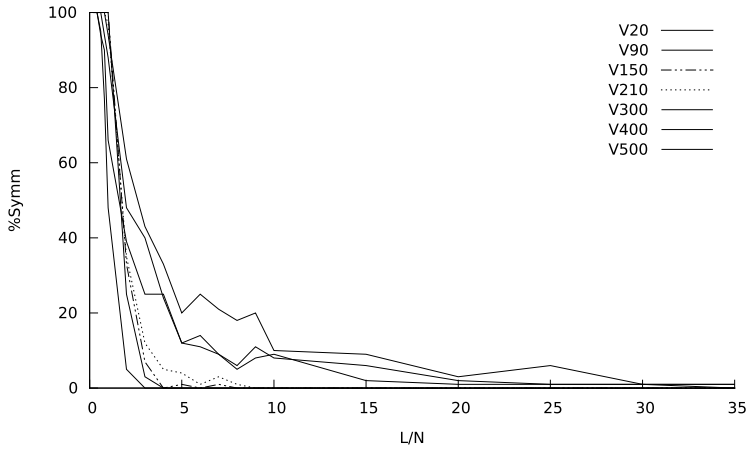


Fig. 2. % of random symmetric instances.

$\bar{\rho}_1 = \langle \rho_{Id}, \rho_{Id}, (p\ \neg p) \rangle,$
$\bar{\rho}_2 = \langle \rho_{Id}, \rho_{Id}, (q\ \neg q) \rangle,$
$\bar{\rho}_3 = \langle \rho_{Id}, (p\ q)(\neg p\ \neg q), (p\ q)(\neg p\ \neg q) \rangle.$

The following theorem is proved in [9].

Theorem 1 *(Correctness). Let φ be a modal CNF formula and $LG^{id}(\varphi) = (N, E, l)$ the graph corresponding to φ. Then every symmetry $\bar{\rho}$ of φ corresponds one-to-one to an automorphism π of $LG^{id}(\varphi)$.*

3.1. Experimental evaluation

In [9], preliminary experimental results showed the existence of symmetries in formulas from the Logics Workbench Benchmark (LWB) for the basic modal logic [19]. The evaluation also showed that detecting symmetries is a relatively inexpensive process. The LWB is a well established, structured test set and has been extensively used to test modal provers. However, most of the formulas became too easy for state of the art provers, even though it still contains instances which remain difficult. This indicates the need for a more thorough testing to better evaluate the behavior of symmetry detection in modal formulas.

To this end we performed an empirical evaluation of the graph construction of Definition 11 to verify the existence of symmetries in modal formulas and the computing costs involved in their detection in larger sets of formulas.

To our knowledge there is no large structured test set for modal logics (equivalent, for example, to the TPTP test set for first-order theorem proving [20]). The standard approach is to use random generators in an attempt to uniformly cover a sufficiently large portion of the spectrum of satisfiable/unsatisfiable formulas (see [12,21] for details). Arguably, the syntactic nature of symmetries makes the approach to testing via random generators ill suited: one would expect symmetries to appear more often in hand-tailored test sets, where formulas follow some predefined pattern. But, as we will see, even in random tests a good number of symmetries still appear.

Fig. 2 shows the results using 19 000 formulas in CNF, generated using the random generator hGen [21]. We fixed the maximum modal depth of the formulas to 3 (this is usually enough to generate sufficiently challenging formulas). Instances are distributed in seven sets, for each set we fixed the number of propositional variables (N) (from 20 to 500) and vary the number of clauses (L) to get different values of the ratio clauses-to-variables (L/N). This ratio is a good indicator of the satisfiability of the formula: formulas with smaller values of L/N are likely to be satisfiable, while formulas with greater values of L/N are often unsatisfiable. Each set contains 100 random instances for 19 different values of the ratio L/N (from 0.2 to 35). Notice that formula size grows with L/N (larger values of L/N are obtained by generating more clauses). The figure shows the percentage of symmetric instances for each value of the ratio L/N. For small values of L/N we find many symmetric instances even in randomly generated formulas. As we increase the value of the ratio, the number of symmetric instances rapidly diminish. This coincides with our expectations: large values of L/N results from a high number of random clauses in the instances, reducing the possibility of symmetries. Summing up, even though random generation can be used to obtain a test set of symmetric formulas, the result will contain mostly short, satisfiable formulas.

In an attempt to obtain more challenging, large structured formulas we used 4113 instances distributed in 23 problem classes from the QBFLib Benchmarks [22]. These are Quantified Boolean Formulas (QBF) encodings of problems mostly from the verification and planning domains. Problems from the QBFLib benchmarks were translated to the basic modal logic using two variations of Ladner's translation [23] that reduces QBFs to modal formulas preserving satisfiability. These translations, named *Collapse1* and *Collapse2* and resulting in the test sets QBFLib-1 and QBFLib-2, respectively, reduce the modal depth of

Table 1
Symmetries in structured test sets.

	#In	#TO	#Sy	#Cl	#SyCl	T_G (sec)	T_S (sec)
LWB	378	0	208	9	6	9.80	1.80
QBFLib-1	4113	20	3874	23	23	7596.48	65012.79
QBFLib-2	4113	20	2680	23	18	6342.06	39370.36

the resulting modal formula yielding smaller formulas than Ladner's original translation (see Appendix A.2 for more details). The resulting formulas are large (the file containing the largest formula in the test set is 200 megabytes long) and they are bound to contain both symmetries from the original problem, and others introduced during the translation into basic modal logic.

Table 1 summarizes the results (including the results for the LWB from [9] for the sake of completeness). All tests were ran on an Intel Core i7 2.93 GHz with 16 GB of RAM with a timeout of 120 seconds for both graph creation and symmetry detection. We used `Bliss` [18] as the graph automorphism detection tool. Columns *#In*, *#TO* and *#Sy* are the number of instances in the test set, the number of timeouts, and the number of instances with at least one symmetry, respectively. Columns *#Cl* and *#SyCl* are the total number of problem classes in the test set and the number of problem classes with at least one symmetric instance, respectively. Columns T_G and T_S are the time in seconds to create the graph and the total time to search for automorphism *for all the instances in the test set*, respectively.

Table 1 shows that many symmetric instances exists in the test sets. The time required to compute the symmetries (graph time plus search time) in the case of LWB is negligible. As we mentioned before, formulas in the QBFLib tests are very large and the associated graphs contains thousands of nodes. The required times, in this test set, are more noticeable, but still quite acceptable. The results show that the detection algorithm is very robust and is able to handle complex test cases. The results for QBFLib highlight how sensitive is symmetry detection to the codification of the formulas. The QBFLib-1 test set have many more symmetric instances than the QBFLib-2 test set. Moreover, all problem classes in QBFLib-1 have symmetric instances, whereas there exists 5 problem classes in the QBFLib-2 that have no symmetric instances. This can be explained by the fact that the translation generating QBFLib-1 uses auxiliary variables (to "mark" the levels in the resulting tree models) while the translation generating QBFLib-2 does not. Therefore, QBFLib-1 formulas have more propositional literals at each modal depth, that might be permuted.

4. Symmetric blocking

In this section we show how the standard labeled tableaux calculus (see, e.g., [24,11]) for the basic modal logic can be controlled with a dynamic blocking mechanism that uses symmetry information from the input formula. We will use standard prefixed tableaux rules and notation which we briefly review for completeness. Let PREF be an infinite, non-empty set of *prefixes*. Here we will set $\mathsf{PREF} = \mathbb{N}$, the set of natural numbers. Given φ a modal CNF formula, C a modal clause and $\sigma \in \mathsf{PREF}$ we call $\sigma{:}\varphi$ and $\sigma{:}C$ *prefixed formulas*. The intended interpretation of a prefixed formula $\sigma{:}F$ is that F holds at the state denoted by σ. Given $\sigma, \sigma' \in \mathsf{PREF}$ we call $\sigma R \sigma'$ an *accessibility statement*. The intended interpretation of $\sigma R \sigma'$ is that the state denoted by σ' is accessible via R from the state denoted by σ.

A tableaux for a formula φ in modal CNF is a tree whose nodes are decorated with prefixed formulas and accessibility statements, such that the root node is $0{:}\varphi$. Moreover, we require that additional nodes in the tree are created according to the following rules, where φ is a modal CNF formula and C a clause.

$$\frac{\sigma{:}\varphi}{\sigma{:}C_i}(\wedge) \quad \text{for all } C_i \in \varphi \qquad \frac{\sigma{:}C}{\sigma{:}l_1 | \ldots | \sigma{:}l_n}(\vee) \quad \text{for all } l_i \in C$$

$$\frac{\sigma{:}\neg\Box C}{\sigma R \sigma',\ \sigma'{:}{\sim}C}(\Diamond)^1 \qquad \frac{\sigma{:}\Box C, \sigma R \sigma'}{\sigma'{:}C}(\Box)$$

[1] ${\sim}C$ is the CNF of the negation of C. The prefix σ' is new in the tableau.

The rules are interpreted as follows: if the antecedents of a rule appear in nodes in a branch of the tableaux, the branch is extended according to the formulas in the consequent. For the case of the $(\vee)$ rule, n immediate successors of the last node of the branch should be created. In all other cases only one successor is created, which is decorated with the indicated formulas. To ensure termination, we require that nodes are created only if they add at least one prefixed formula or accessibility statement that was not already in the branch. Moreover, the $(\Diamond)$ rule can only be applied once to each formula of the form $\sigma{:}\neg\Box C$ in a branch. This is called the *standard blocking condition*. A branch is *closed* if both $\sigma{:}p$ and $\sigma{:}\neg p$ appear as labels in some node of the branch, and it is *open* otherwise. A branch is *saturated* if no rule can be further applied in the branch. The conditions imposed on rule applications lead to saturation after a finite number of steps.

Let $\mathsf{Tab}(\varphi)$ be the set of tableaux for φ whose branches are all saturated. The following classical result establishes that the tableaux calculus we just defined is a decision method for satisfiability of formulas in the basic modal logic (see [25] for details).

Theorem 2. *For any formula φ in modal CNF, any $\mathsf{T} \in \mathsf{Tab}(\varphi)$ is finite. Moreover T has a saturated open branch if and only if φ is satisfiable.*

A different blocking technique called Pattern-based blocking (PBB) was introduced in [26] and resulted in improved performance. The pattern $P(\sigma{:}\Diamond\varphi)$ of $\sigma{:}\Diamond\varphi$ is the set of formulas consisting of $\sigma{:}\Diamond\varphi$ together with all $\sigma{:}\Box\theta$ formulas in the branch. Once the $(\Diamond)$ rule is applied to $\sigma{:}\Diamond\varphi$, $P(\sigma{:}\Diamond\varphi)$ is marked as expanded. Moreover, a pattern P is considered expanded if there is an expanded pattern Q such that $P \subseteq Q$. PBB restricts the applicability of the $(\Diamond)$ rule to formulas whose patterns are not yet expanded on the branch. PBB is also sound, complete and ensures termination.

In what follows, we will strengthen these blocking conditions, further restricting the application of the $(\Diamond)$ rule so that it can be applied to a $\sigma{:}\neg\Box C$ formula only if it has not been applied to symmetric formulas before. We will present three different symmetric blocking conditions: Symmetric Blocking 1 (SB1), Symmetric Blocking 2 (SB2) and Symmetric Pattern-based Blocking (SPBB). The first one was presented already in [27]. These restrictions cannot affect soundness or termination of the (already sound and terminating) calculus. We will prove that they do not affect completeness either, i.e., Theorem 2 holds also under these new symmetric blocking conditions.

4.1. Preliminaries

Definition 12 *(Depth of a prefix, permutation of a prefixed formula).* Let $\mathsf{T} \in \mathsf{Tab}(\varphi)$, and let Θ be a branch of T. Each prefix σ in T, except 0, is introduced by the $(\Diamond)$ rule, and these applications define a unique path $0R\sigma_1 R \dots R\sigma$ between 0 and any other prefix. Let $\mathsf{depth}(\sigma)$ denote the length of this path (in particular, $\mathsf{depth}(0) = 0$). Given a prefixed formula $\sigma{:}\varphi$ and a layered permutation $\bar{\rho}$, define $\bar{\rho}(\sigma{:}\varphi) = \sigma{:}\bar{\rho}_{\mathsf{depth}(\sigma)+1}(\varphi)$.

We will need to identify the set of $\Box$-formulas occurring at a given prefix in a branch of a tableau.

Definition 13 *(Set of $\Box$-formulas occurring at a prefix).* Let φ be a modal formula, $\mathsf{T} \in \mathsf{Tab}(\varphi)$, Θ be a branch of T and σ a prefix in Θ. By $\Gamma(\sigma) = \{\psi \mid \sigma{:}\Box\psi \in \Theta\}$ we denote the set of $\Box$-formulas occurring at prefix σ.

To prove completeness of the calculus with symmetric blocking we rely on the intuition that we can extend an incomplete model $\mathcal{M}^\Theta$, built from a saturated open branch Θ to a complete model even when symmetric blocking was used.

Definition 14 *(Model $\mathcal{M}^\Theta$ associated to a branch Θ).* Given an open saturated branch Θ of a tableaux $\mathsf{T} \in \mathsf{Tab}(\varphi)$, we define the model $\mathcal{M}^\Theta = \langle 0, W^\Theta, R^\Theta, V^\Theta\rangle$ as:

$$\begin{aligned} W^\Theta &= \{\sigma \mid \sigma \text{ is a prefix on } \Theta\} \\ R^\Theta &= \{(\sigma,\sigma') \mid \sigma,\sigma' \in W^\Theta \text{ and } \sigma R\sigma' \in \Theta\} \\ V^\Theta(\sigma) &= \{p \mid \sigma{:}p \in \Theta\}. \end{aligned}$$

Notice that $\mathcal{M}^\Theta$ is always a tree rooted at 0.

Definition 15 *(Rooted sub-model).* Given a model $\langle w, W, R, V\rangle$ and an element $v \in W$, let $W[v]$ denote the set of all elements that are reachable from v by the reflexive and transitive closure of R. Let $\mathcal{M}[v] = \langle v, W[v], R_{\restriction W[v]}, V_{\restriction W[v]}\rangle$ denote the *sub-model of $\mathcal{M}$ rooted at v*.

We already introduced the $\bar{\rho}$-image of a model in Definition 9. We now introduce a variant of that model construction.

Definition 16 *($\bar{\rho}$-left image).* Given a pointed tree model $\mathcal{M} = \langle w, W, R, V\rangle$, a layered permutation $\bar{\rho}$, and disjoint sets $L, R \subseteq \mathsf{PROP}$. Let $\mathcal{L} = \mathcal{M}_{\restriction L} = \langle w, W, R, V_{\restriction L}\rangle$ and $\mathcal{R} = \mathcal{M}_{\restriction R} = \langle w, W, R, V_{\restriction R}\rangle$. Define the *$\bar{\rho}$-left image of $\mathcal{M}$* as the model $\mathcal{M}_{\bar{\rho},L,R} = \mathcal{L}_{\bar{\rho}} \cup \mathcal{R} = \langle w, W, R, V_{\mathcal{L}_{\bar{\rho}}\cup\mathcal{R}}\rangle$ where $\mathcal{L}_{\bar{\rho}}$ is the $\bar{\rho}$-image of $\mathcal{L}$ and $V_{\mathcal{L}_{\bar{\rho}}\cup\mathcal{R}}(v) = V_{\mathcal{L}_{\bar{\rho}}}(v) \cup V_{\mathcal{R}}(v)$ for all $v \in W$.

Definition 17 *($M[\Sigma_\Theta]$).* Given Θ a saturated open branch, let Σ_Θ be the set of tuples $(\sigma, \psi, \bar{\rho}, b)$ where σ is a prefix added to Θ by the application of the $(\Diamond)$ rule to a $\neg\Box$-formula ψ that has a symmetric $\neg\Box$-formula $\bar{\rho}(\psi)$ blocked using the blocking condition $b \in \{SB1, SB2, SPBB\}$. We define the set of models $M[\Sigma_\Theta]$ as

$$\begin{aligned} M[\Sigma_\Theta] = \{&\mathcal{M}^\Theta[\sigma]_{\bar{\rho}} \mid (\sigma,\psi,\bar{\rho},b) \in \Sigma_\Theta, b \in \{SB2, SPBB\}\} \\ &\cup \{\mathcal{M}^\Theta[\sigma]_{\bar{\rho},\mathsf{PROP}(\psi),\mathsf{PROP}(\Gamma(\sigma))} \mid (\sigma,\psi,b) \in \Sigma_\Theta, b \in \{SB1\}\}, \end{aligned}$$

where $\mathcal{M}^\Theta[\sigma]_{\bar{\rho}}$ and $\mathcal{M}^\Theta[\sigma]_{\bar{\rho},\mathsf{PROP}(\psi),\mathsf{PROP}(\Gamma(\sigma))}$ are the $\bar{\rho}$-image and the $\bar{\rho}$-left image of model $\mathcal{M}^\Theta[\sigma]$ respectively.

Intuitively, $M[\Sigma_\Theta]$ is the set of "symmetric" sub-models that we need to glue to the model $\mathcal{M}^\Theta$ to complete it. Also, notice that models in $M[\Sigma_\Theta]$ are constructed differently depending on the blocking condition (SB1, SB2 and SPBB) used when blocking a $\neg\Box$-formula.

Definition 18 *(Symmetric extension).* Given a saturated open branch Θ, the associated model $\mathcal{M}^\Theta = \langle 0, W^\Theta, R^\Theta, V^\Theta\rangle$ and a set of symmetric pointed sub-models $M[\Sigma_\Theta]$, the *symmetric extension of* $\mathcal{M}^\Theta$ is the model $\mathcal{M}^\Theta_{\bar\rho} = \langle 0, W^\Theta_{\bar\rho}, R^\Theta_{\bar\rho}, V^\Theta_{\bar\rho}\rangle$ where:

$$\begin{aligned} W^\Theta_{\bar\rho} &= W^\Theta \uplus \biguplus W \\ R^\Theta_{\bar\rho} &= R^\Theta \uplus \biguplus R \cup \{(\sigma, \tau_{\sigma'}) \mid (\sigma,\sigma') \in R^\Theta\} \\ V^\Theta_{\bar\rho} &= V^\Theta \uplus \biguplus V \end{aligned}$$

for all $\langle \tau_{\sigma'}, W, R, V\rangle \in M[\Sigma_\Theta]$, where $\tau_{\sigma'}$ is the element corresponding to σ' in the disjoint union.

Note that we are gluing the symmetric sub-models to the original model by adding an edge from the element σ to the root $\tau_{\sigma'}$ of the symmetric sub-model if there is an edge from σ to the root σ' of the original sub-model.

The following proposition is well known (see, e.g., [10]) and states that the evaluation of a formula in a model only depends on the propositional variables appearing in the formula.

Proposition 4. *Let φ be a modal formula, then $\langle w, W, R, V\rangle \models \varphi$ iff $\langle w, W, R, V_{\restriction \mathsf{PROP}(\varphi)}\rangle \models \varphi$.*

4.2. Completeness

We will now start the completeness proof. The proof will depend on the following Blocking Lemma that will be established later.

Blocking Lemma. *Let φ be a modal CNF formula, $\bar\rho$ a layered symmetry of φ and Θ a saturated open branch of $\mathsf{T} \in \mathsf{Tab}(\varphi)$. Let σ be a prefix such that $\sigma{:}\neg\Box\psi \in \Theta$ and let $\sigma{:}\bar\rho(\neg\Box\psi) \in \Theta$ be a blocked formula using any of SB1, SB2 or SPBB. Then $\bar\rho(\psi) \wedge \bigwedge \Gamma(\sigma)$ is satisfiable.*

We start by establishing the correspondence between membership of a prefixed formula in the branch Θ and truth in the symmetric extension model built from it.

Lemma 1. *Let Θ be a saturated open branch of a tableau $\mathsf{T} \in \mathsf{Tab}(\varphi)$ and $\bar\rho$ a symmetry of φ. For any formula $\sigma{:}\psi \in \Theta$ we have that $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \psi$.*

Proof. The proof is by induction on both the modal depth and the size of ψ.

$[\psi = p]$: By definition, $\sigma \in V^\Theta_{\bar\rho}(p)$. This implies $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models p$.

$[\psi = \neg p]$: Since Θ is open, $\sigma{:}p \notin \Theta$. Thus $\sigma \notin V^\Theta_{\bar\rho}(p)$, which implies $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \neg p$.

$[\psi = \chi \wedge \theta]$ and $[\psi = \chi \vee \theta]$ are trivial, by application of the corresponding tableau rules and the induction hypothesis.

$[\psi = \neg\Box\theta]$: We have to consider two cases: a) $\neg\Box\theta$ has been expanded by the application of the $(\Diamond)$ rule. By saturation of $(\Diamond)$, $\sigma R\sigma'$, $\sigma'{:}\sim\theta \in \Theta$. By definition of $R^\Theta_{\bar\rho}$ and induction hypothesis (which can be applied because $\sim\theta$ has smaller modal depth): $(\sigma,\sigma') \in R^\Theta_{\bar\rho}$ and $\mathcal{M}^\Theta_{\bar\rho}, \sigma' \models \sim\theta$ and, hence, $\mathcal{M}^\Theta_{\bar\rho}, \sigma' \models \neg\theta$. Combining this, we obtain $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \neg\Box\theta$, as required. b) $\neg\Box\theta$ has been blocked by the application of symmetric blocking. In this case, $\neg\Box\theta = \bar\rho(\neg\Box\chi) = \neg\Box\bar\rho(\chi)$. By saturation of $(\Diamond)$ we have that $\sigma R\sigma'$, $\sigma'{:}\sim\chi \in \Theta$. Moreover, we have that $(\sigma,\sigma') \in R^\Theta$ and that $\mathcal{M}^\Theta, \sigma' \models \neg\chi$. By definition of the symmetric extension of $\mathcal{M}^\Theta$ we have that $(\sigma, \tau_{\sigma'}) \in R^\Theta_{\bar\rho}$ and $\mathcal{M}^\Theta_{\bar\rho}, \tau_{\sigma'} \models \neg\bar\rho(\chi)$. Which implies that $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \neg\Box\bar\rho(\chi) = \neg\Box\theta$.

$[\varphi = \Box\theta]$: If there is no state σ' such that $(\sigma,\sigma') \in R^\Theta_{\bar\rho}$ then this holds trivially. Otherwise, let σ' be such that $(\sigma,\sigma') \in R^\Theta_{\bar\rho}$. By definition of $R^\Theta_{\bar\rho}$ it must be the case that $\sigma{:}\neg\Box\chi \in \Theta$ and $\sigma R\sigma' \in \Theta$. We must consider two cases: a) if $\sigma{:}\neg\Box\chi$ has no symmetric counterpart, i.e., it is not blocking a formula $\sigma{:}\neg\Box\bar\rho(\chi)$ then, given that $\sigma{:}\Box\theta \in \Theta$, by saturation of $(\Box)$, we have that $\sigma'{:}\theta \in \Theta$. By inductive hypothesis, we have that $\mathcal{M}^\Theta_{\bar\rho}, \sigma' \models \theta$. From this it follows that $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \Box\theta$ as required. b) If it is the case that $\sigma{:}\neg\Box\chi$ is blocking $\sigma{:}\neg\Box\bar\rho(\chi)$, then, by the definition of the symmetric extension $\mathcal{M}^\Theta_{\bar\rho}$ and the Blocking Lemma, we have that $(\sigma, \tau_{\sigma'}) \in R^\Theta_{\bar\rho}$ and $\mathcal{M}^\Theta_{\bar\rho}, \tau_{\sigma'} \models \bar\rho(\chi) \wedge \Gamma(\sigma)$. Given that $\theta \in \Gamma(\sigma)$ then, $\mathcal{M}^\Theta_{\bar\rho}, \tau_{\sigma'} \models \theta$. From what it follows that $\mathcal{M}^\Theta_{\bar\rho}, \sigma \models \Box\theta$ as required. □

Theorem 3. *The tableau calculus with symmetric blocking is complete.*

Proof. Let Θ be an open saturated branch of the tableau $\mathsf{T} \in \mathsf{Tab}(\varphi)$. Since $0{:}\varphi \in \Theta$, by Lemma 1 φ is satisfiable. □

4.3. Symmetric blocking conditions

It rest only to define the symmetric blocking conditions and prove the Blocking Lemma.

Symmetric Blocking 1 (SB1): Let $\bar{\rho}$ be a symmetry of φ, and let Θ be a branch in a tableau of φ. The rule ($\Diamond$) cannot be applied to $\sigma{:}\bar{\rho}(\neg\Box\psi)$ on Θ if it has been applied to $\sigma{:}\neg\Box\psi$ and $\mathsf{PROP}(\bar{\rho}(\neg\Box\psi)) \cap \mathsf{PROP}(\Gamma(\sigma)) = \emptyset$.

Symmetric Blocking 2 (SB2): Let $\bar{\rho}$ be a symmetry of φ, and let Θ be a branch in a tableau of φ. The rule ($\Diamond$) cannot be applied to $\sigma{:}\bar{\rho}(\neg\Box\psi)$ on Θ if it has been applied to $\sigma{:}\neg\Box\psi$ and $\bar{\rho}(\Gamma(\sigma)) = \Gamma(\sigma)$.

Symmetric Pattern-Based Blocking (SPBB): Let $\bar{\rho}$ be a symmetry of φ, and let Θ be a branch in a tableau of φ. The rule ($\Diamond$) cannot be applied to $\sigma{:}\bar{\rho}(\neg\Box\psi)$ on Θ if the pattern $P(\sigma'{:}\bar{\rho}(\neg\Box\varphi))$ has been already expanded, for σ' a prefix in Θ, and $\bar{\rho}(\Gamma(\sigma)) = \Gamma(\sigma)$.

Note that these symmetric blocking conditions are dynamic: after being blocked, a $\neg\Box$-formula can be re-scheduled if the blocking condition fails in an expansion of the current branch. This can happen because the set $\Gamma(\sigma)$ increases monotonically as the tableau advances.

Blocking Lemma 1. *Let φ be a modal CNF formula, $\bar{\rho}$ a layered symmetry of φ and Θ a saturated open branch using SB1 of $\mathsf{T} \in \mathsf{Tab}(\varphi)$. Let σ be a prefix such that $\sigma{:}\neg\Box\psi \in \Theta$ and let $\sigma{:}\bar{\rho}(\neg\Box\psi) \in \Theta$ be a blocked formula using any of SB1, SB2 or SPBB. Then $\bar{\rho}(\psi) \wedge \bigwedge \Gamma(\sigma)$ is satisfiable.*

Proof. We divide the proof into cases depending on the used blocking condition:

[SB1]: Given that Θ is a saturated open branch, there exists σ' such that $\{\sigma R\sigma'\} \cup \{\sigma'{:}\psi\} \cup \{\sigma'{:}\psi \mid \psi \in \Gamma(\sigma)\} \in \Theta$. From Θ we can construct a model $\mathcal{M}^\Theta = \langle W^\Theta, R^\Theta, V^\Theta\rangle$ such that $\mathcal{M}^\Theta, 0 \models \varphi$ and, in particular, $\mathcal{M}^\Theta, \sigma' \models \psi \wedge \bigwedge \Gamma(\sigma)$ with $(\sigma, \sigma') \in R^\Theta$.

Let $\mathcal{M}^\Theta[\sigma'] = \langle \sigma', W', R', V'\rangle$ be the sub-model rooted at σ' with $W' = W^\Theta[\sigma']$, $R' = R^\Theta_{\restriction W^\Theta[\sigma']}$ and $V' = V^\Theta_{\restriction W^\Theta[\sigma']}$. Then $\mathcal{M}^\Theta[\sigma'] \models \psi \wedge \bigwedge \Gamma(\sigma)$. Let $\mathcal{N} = \mathcal{M}^\Theta[\sigma']_{\restriction \mathsf{PROP}(\psi)} = \langle \sigma', W', R', V'_{\mathcal{N}}\rangle$ and $\mathcal{R} = \mathcal{M}^\Theta[\sigma']_{\restriction \mathsf{PROP}(\Gamma(\sigma))} = \langle \sigma', W', R', V'_{\mathcal{R}}\rangle$. By Proposition 4 $\mathcal{N} \models \psi$ and $\mathcal{R} \models \bigwedge \Gamma(\sigma)$.

By Propositions 1 and 2, $\mathcal{N}_{\bar{\rho}} \models \bar{\rho}(\psi)$. Finally, let $\mathcal{U} = \mathcal{N}_{\bar{\rho}} \cup \mathcal{R} = \langle \sigma, W', R', V''\rangle$ where $V''(w) = V'_{\mathcal{N}_{\bar{\rho}}}(w) \cup V'_{\mathcal{R}}(w)$ for all $w \in W'$. By the symmetric blocking condition we know that $\mathsf{PROP}(\bar{\rho}(\psi)) \cap \mathsf{PROP}(\Gamma(\sigma)) = \emptyset$ and therefore $L_{V'_{\mathcal{N}_{\bar{\rho}}}(w)} \cap L_{V'_{\mathcal{R}}(w)} = \emptyset$ for all $w \in W'$. It follows that no contradiction will arise when doing $V'_{\mathcal{N}_{\bar{\rho}}}(w) \cup V'_{\mathcal{R}}(w)$ and hence that the valuation function $V''(w)$ is well defined. Notice that $\mathcal{U}$ is the $\bar{\rho}$-left image of $\mathcal{M}^\Theta[\sigma']$.

It rest to prove that $\mathcal{U} \models \bar{\rho}(\psi) \wedge \bigwedge \Gamma(\sigma)$. Consider $\mathcal{U}_{\restriction \mathsf{PROP}(\bar{\rho}(\psi))}$, by construction of $\mathcal{U}$, we know that $\mathcal{U}_{\restriction \mathsf{PROP}(\bar{\rho}(\psi))} = \mathcal{N}_{\bar{\rho}}$ and that $\mathcal{N}_{\bar{\rho}} \models \bar{\rho}(\psi)$. By Proposition 4, $\mathcal{U} \models \bar{\rho}(\psi)$. That $\mathcal{U} \models \bigwedge \Gamma(\sigma)$ follows by the same argument using $\mathcal{R}$.

[SB2]: Since Θ is a saturated open branch, there exists σ' such that $\{\sigma R\sigma'\} \cup \{\sigma'{:}\psi\} \cup \{\sigma'{:}\psi \mid \psi \in \Gamma(\sigma)\} \subseteq \Theta$. We can construct a model $\mathcal{M}^\Theta = \langle W^\Theta, R^\Theta, V^\Theta\rangle$ from Θ such that $\mathcal{M}^\Theta, 0 \models \varphi$ and, in particular, $\mathcal{M}^\Theta, \sigma' \models \psi \wedge \bigwedge \Gamma(\sigma)$ with $(\sigma, \sigma') \in R^\Theta$.

Now consider the model $\mathcal{M}^\Theta_{\bar{\rho}}$. By Propositions 1 and 2, it follows that $\mathcal{M}^\Theta_{\bar{\rho}}, \sigma' \models \bar{\rho}(\psi \wedge \bigwedge \Gamma(\sigma)) = \bar{\rho}(\psi) \wedge \bigwedge \bar{\rho}(\Gamma(\sigma))$. By SB2, $\bar{\rho}(\Gamma(\sigma)) = \Gamma(\sigma)$, therefore, $\mathcal{M}^\Theta_{\bar{\rho}}, \sigma' \models \bar{\rho}(\psi) \wedge \bigwedge \Gamma(\sigma)$.

[SPBB]: Same argument as for the SB2 case, since SPBB also requires that $\bar{\rho}(\Gamma(\sigma)) = \Gamma(\sigma)$. □

Notice that SB2 can be seen as particular case of SPBB in which the search for a matching pattern is restricted to the current prefix, i.e., the prefix of the formula being blocked. Moreover, it is easy to observe that SB2 is subsumed by SPBB, i.e., every SB2-blocked formula is a SPBB-blocked formula. The advantage of using SB2 over SPBB is that the former could be implemented more efficiently than the latter, since the search for a symmetric formula is restricted to the current prefix.

5. Experimental evaluation

We now empirically evaluate the effect that symmetric blocking has on modal benchmarks.

5.1. Implementation

We implemented the symmetric blocking conditions SB1, SB2 and SPBB in `HTab` [28], a tableaux prover for the hybrid logic $\mathcal{H}(:, \mathsf{E}, \mathsf{D}, \Diamond^-, \downarrow)$ with reflexive, transitive and symmetric modalities. `HTab` includes a series of optimizations that are

Table 2
Tested HTab configurations.

Configuration	Description
HTab	The original prover with the default configuration
HTab_{SB1}	HTab + SB1
HTab_{SB2}	HTab + SB2
$\text{HTab}^{=}$	HTab + SPBB with exact match
$\text{HTab}^{\subseteq}$	HTab + SPBB with subset match
$\text{HTab}^{=}_{\text{SB2}}$	HTab + SB2 + SPBB with exact match
$\text{HTab}^{\subseteq}_{\text{SB2}}$	HTab + SB2 + SPBB with subset match

enabled by default, namely, pattern-based blocking [29], semantic branching [30], dependency-directed backtracking [30], lazy branching [26], unit propagation [31] and eager unit propagation [28].

The implementation of SB1 and SB2 is as follows: whenever there is a formula $\sigma{:}\neg\Box\varphi$ scheduled for expansion, the solver checks if there is a symmetric formula $\sigma{:}\bar{\rho}(\neg\Box\varphi)$ already expanded. If this is the case, it blocks the formula $\sigma{:}\neg\Box\varphi$ and continues with the application of the remaining rules. The solver only verifies the blocking condition if it gets a saturated open branch. If the blocking condition holds for all blocked formulas the solver terminates. Otherwise it reschedules formulas for further expansion.

The implementation of SPBB is as follows: whenever there is a formula $\sigma{:}\neg\Box\varphi$ scheduled for expansion, the solver computes its symmetric pattern $P(\sigma'{:}\bar{\rho}(\neg\Box\varphi))$ and verifies if there exists a matching pattern that was already expanded. If that is the case the solver blocks the formula $\sigma{:}\neg\Box\varphi$ and continues with the application of the remaining rules. If the solver finds a saturated open branch, then it verifies the blocked formulas using the blocking condition. If the blocking condition holds, it terminates. Otherwise it reschedules formulas for further expansion. A matching pattern can be an *exact* match, i.e., it is identical to the original pattern, or a *subset* match, i.e., it *contains* the symmetric pattern.

If more than one blocking condition are enabled simultaneously, those enabled are evaluated in the following order: SB1 is tried first, then SB2 and finally SPBB.

To investigate the effects of symmetric blocking on modal benchmarks, we tested different solver configurations combining SB1, SB2 and SPBB. As we will see, in our tests SB2 generally outperformed SB1, hence we only considered the combination of SB2 and SPBB. Table 2 describes the configurations used.

5.2. Results

Our benchmark contains formulas from the LWB (207 formulas from 5 problem classes) and QBFLib-2 (172 formulas from 15 problem classes) — from now we will refer to these as just QBFLib — benchmarks described in Section 3. All formulas contain at least one non-trivial symmetry. Tests were run on a cluster using 4 Intel Xeon E5 2680 v2 nodes, with 64 GB of RAM. Timeout was set to 180 seconds.

Table 3 presents the results for the 7 solver configurations. For each configuration, the table shows the number of instances solved (row *Solved*), the total time, in seconds, required to test all the formulas, including timeouts and symmetry computation time when pertinent (row *Time*), the number of satisfiable and unsatisfiable instances solved (rows *Sat* and *Unsat* respectively) and the number of timeouts (row *TO*). For each configuration, we present the values corresponding to each test set and the total for the complete benchmark. The table shows that all configurations using SB1, SB2 and/or SPBB outperform HTab. HTab_{SB2} emerges as the best performing configuration. Configurations $\text{HTab}^{=}$ and $\text{HTab}^{\subseteq}$, that only use SPBB, show a similar performance to HTab_{SB2}: they outperform HTab and HTab_{SB1}, but solved 1 problem less than HTab_{SB2} and required more time in general. Configurations $\text{HTab}^{=}_{\text{SB2}}$ and $\text{HTab}^{\subseteq}_{\text{SB2}}$, combining SB2 and SPBB, outperform HTab and HTab_{SB1}, but they do not perform as well as the configurations in which SB2 and SPBB are used separately. Table 3 shows also that SPBB with exact matching ($\text{HTab}^{=}$) performs slightly better than SPBB with subset matching ($\text{HTab}^{\subseteq}$).

Figs. 3a and 3b show scatter plots comparing the runtimes of HTab_{SB2} against HTab and HTab_{SB1} respectively. The *x* axis gives the running times of HTab in Fig. 3a and of HTab_{SB1} in Fig. 3b, whereas the *y* axis gives the running times of HTab_{SB2} in both figures. Each point represents an instance and its horizontal and vertical coordinates represent the time necessary to solve it in seconds. Points on the rightmost and topmost edges represent timeout. Notice that a logscale is used, so that gain or degradation to the far right and far top are exponentially more relevant.

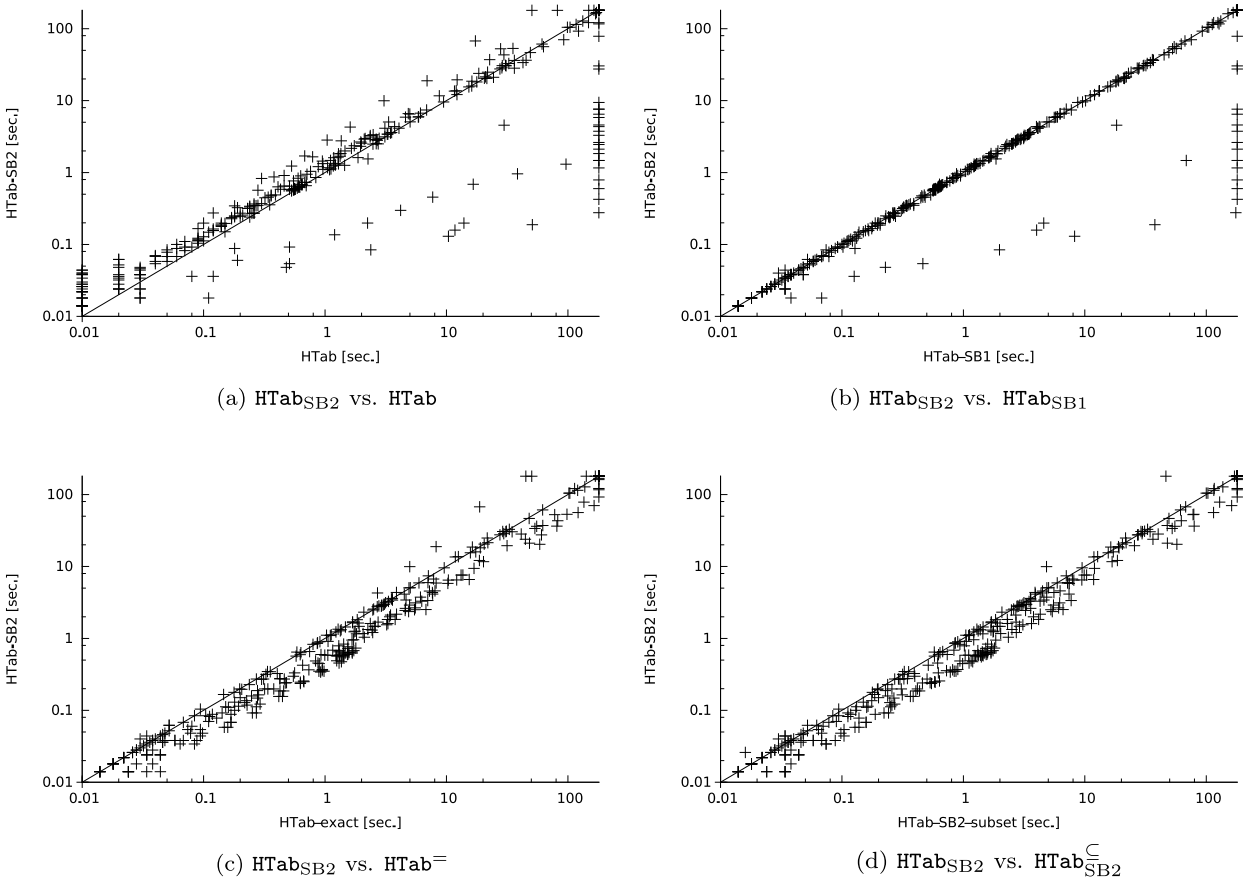
Fig. 3a shows that for many instances (approximately 31% of all instances) HTab_{SB2} reports an important performance gain with respect to HTab. The remaining instances report some degradation, in most cases negligible. Performance degradation is caused by the additional overhead imposed by the blocking mechanism in instances that never trigger symmetric blocking or in those cases where, after being triggered, the blocking condition does not hold, resulting in the final expansion of the blocked formula.

Fig. 3b compares the performance of HTab_{SB2} and HTab_{SB1}. It shows that HTab_{SB2} consistently outperforms HTab_{SB1}: many instances show an impressive performance gain, and no instance show degradation.

Figs. 3c and 3d show scatter plots comparing the runtimes of HTab_{SB2} against $\text{HTab}^{=}$ (the best configuration using SPBB standalone) and $\text{HTab}^{\subseteq}_{\text{SB2}}$ (the best configuration combining SB and SPBB) respectively. Both figures clearly show that

Table 3
Results on the complete benchmark.

		HTab	HTab_{SB1}	HTab_{SB2}	$\text{HTab}^{=}$	$\text{HTab}^{\subseteq}$	$\text{HTab}^{=}_{\text{SB2}}$	$\text{HTab}^{\subseteq}_{\text{SB2}}$
Solved	QBFLib	165	165	**168**	166	166	164	164
	LWB	175	183	193	**194**	**194**	**194**	**194**
	Total	340	348	**361**	360	360	358	358
Time	QBFLib	3712	3764	**3266**	3806	3820	3902	3895
	LWB	6292	4917	**2751**	2820	2831	2827	2797
	Total	10 004	8681	**6017**	6626	6651	6729	6692
Sat	QBFLib	60	61	**64**	62	62	62	62
	LWB	90	91	**102**	**102**	**102**	**102**	**102**
	Total	150	152	**166**	164	164	164	164
Unsat	QBFLib	**105**	104	104	104	104	102	102
	LWB	85	**92**	91	**92**	**92**	**92**	**92**
	Total	190	**196**	195	**196**	**196**	194	194
TO	QBFLib	7	7	**4**	6	6	8	8
	LWB	32	24	14	**13**	**13**	**13**	**13**
	Total	39	31	**18**	19	19	21	21



(a) HTab_{SB2} vs. HTab

(b) HTab_{SB2} vs. HTab_{SB1}

(c) HTab_{SB2} vs. $\text{HTab}^{=}$

(d) HTab_{SB2} vs. $\text{HTab}^{\subseteq}_{\text{SB2}}$

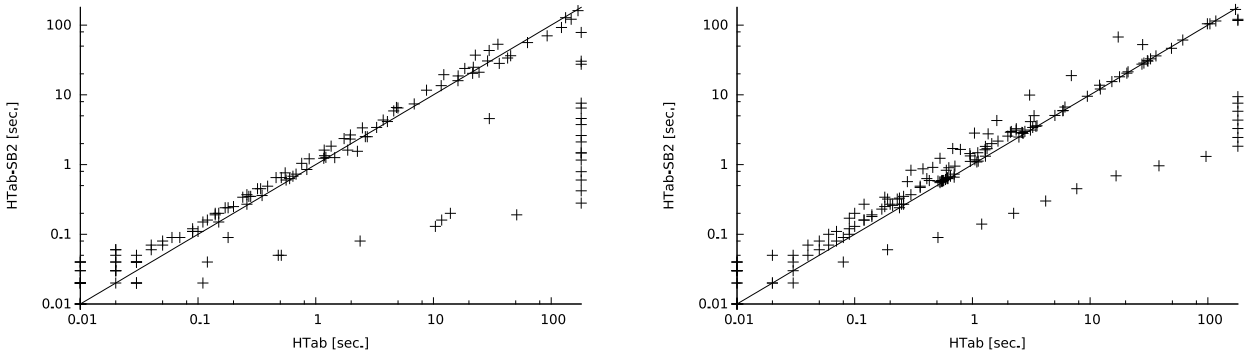
Fig. 3. Performance on the complete benchmark.

HTab_{SB2} consistently outperforms $\text{HTab}^{=}$ and $\text{HTab}^{\subseteq}_{\text{SB2}}$, except for a few isolated instances in which $\text{HTab}^{=}$ and $\text{HTab}^{\subseteq}_{\text{SB2}}$ perform better.

Table 4 shows information about the application of symmetric blocking on satisfiable and unsatisfiable instances in HTab_{SB2}. Column *Solved* is the number of solved instances, column *#ITrig* is the number of instances that triggered SB2 at least once, columns *#TTrig* and *#Resch* are the total number of times that SB2 is triggered and the number of times that blocked formulas had to be rescheduled because the blocking condition failed.

Table 4
Applications of SB in $\texttt{HTab}_{\texttt{SB2}}$.

	Solved	#ITrig	#TTrig	#Resch
Sat	166	89	2031	1192
Unsat	195	76	814	71



(a) Sat (b) Unsat

Fig. 4. Performance of $\texttt{HTab}_{\texttt{SB2}}$.

The table shows that for satisfiable instances, SB2 triggered many times but in approximately half of the cases the blocking condition failed later in the branch causing the blocked formulas to be rescheduled for expansion. Despite this, the scatter plot in Fig. 4a shows that there is still an overall improvement and that for many instances performance gains are significant. It also shows that performance degradation, when it happens, is almost negligible. These observations tell us that, for satisfiable instances, even in cases where the blocking condition does not hold, delaying the processing of symmetric formulas might be beneficial since the branch has more information available that can avoid branching or close the branch more rapidly.

For unsatisfiable instances, Table 4 shows that SB2 triggered less often than for satisfiable instances, however, the blocking condition holds most of the times (recall that the blocking condition is not validated if the branch closes). As expected, Fig. 4b shows great improvement for several instances. However, performance losses are also more noticeable for some instances. A possible explanation is the following: if the blocked formula plays no role in the unsatisfiability of the problem, blocking it avoids unnecessary work resulting in a performance gain. If it plays a role in the unsatisfiability, the solver might be forced to process formulas that would not be processed otherwise, resulting in a performance degradation.

Most of the performance gains obtained with $\texttt{HTab}_{\texttt{SB2}}$ come from formulas in particular problem classes (e.g., the `k_branch` problem class in LWB, or the `Wmiforward` problem class in QBFLib). This again indicates that the effectiveness of symmetric blocking highly depends on the problem class at hand.

Tables 5 and 6 show detailed data concerning the application of symmetric blocking in $\texttt{HTab}_{\texttt{SB2}}$, for each problem class in the QBFLib and LWB test sets respectively. Column *#Inst* is the number of instances in the class, column *#ITrig* is the number of instances that triggered SB2 at least once, columns *#TTrig* and *#Resch* are the total number of times that SB2 is triggered and the number of times that blocked formulas had to be rescheduled due to the blocking condition not holding. Columns $T_{\texttt{HTab}}$ and $T_{\texttt{SB2}}$ are the total time, in seconds, required to process all instances in the class for `HTab` and $\texttt{HTab}_{\texttt{SB2}}$ respectively, including timeouts and symmetry computation (for $\texttt{HTab}_{\texttt{SB2}}$). Columns $n^{100}_{\texttt{HTab}}$ and $n^{100}_{\texttt{SB2}}$ in Table 6, report the number of instances solved within a timeout of 100 seconds by `HTab` and $\texttt{HTab}_{\texttt{SB2}}$ respectively. Table 5 confirms that the application of SB2 depends on the problem class at hand: while all problem classes contain only symmetric instances, only 8 from the 15 problem classes present instances that triggered SB2. This suggest that SB2 only uses a portion of the available symmetries. In some classes, SB2 led to important performances gains even when the blocking condition fails, while in other classes (e.g., `Toilet_C`), SB2 led to performance degradation. Table 6 shows a similar behavior with SB2 triggering only in 4 problem classes. SB2 triggers in `k_path`, but the problems in this class are too easy for this to result in a noticeable performance gain. On the other hand, as mentioned previously, $\texttt{HTab}_{\texttt{SB2}}$ shows a huge performance gain on `k_branch` problems requiring just 1.5% of the time required by `HTab`.

To put the previous results in perspective, we also compared the relative performance of symmetric blocking against two state-of-the-art solvers: `Spartacus v1.1.3` [26] and `Fact++ v1.6.3` [32]. `Spartacus` is a tableaux prover for hybrid logic with global modalities and it was the first prover to implement pattern-based blocking. `Fact++` is a description logic solver that currently support OWL-DL and (partially) OWL 2 DL.

Table 7 shows the results for $\texttt{HTab}_{\texttt{SB2}}$, `Spartacus` and `Fact++` on the complete benchmarks. `Spartacus` is ran over the same formulas used for $\texttt{HTab}_{\texttt{SB2}}$. For `Fact++`, we first translated all formulas to TBoxes (see [33] for details), and then used `Fact++` to check satisfiability of those TBoxes. The table shows that, in the benchmark under consideration, $\texttt{HTab}_{\texttt{SB2}}$

Table 5
Performance of $\texttt{HTab}_{\texttt{SB2}}$ in the QBFLib test set.

Class	#Inst	#ITrig	#TTrig	#Resch	T_{HTab}	T_{SB2}
Toilet_A	20	0	0	0	355	**354**
Toilet_C	62	59	376	229	**1564**	1873
Toilet_G	6	6	34	1	**2**	**2**
blackbox-01X-QBF	25	1	14	0	245	**195**
StrategicCompanies	20	0	0	0	**12**	**12**
Wmiforward	9	8	302	0	566	**212**
Z4ml	8	0	0	0	**11**	**11**
Incrementer-encoder	4	0	0	0	**11**	12
Traffic-light	3	0	0	0	73	**72**
Chain	1	1	12	0	180	**79**
Toilet	5	4	14	14	48	**39**
Connect2	5	0	0	0	229	**225**
Tree	2	2	513	511	**210**	126
BMC	1	0	0	0	**28**	**28**
Qshifter	1	1	516	508	180	**28**

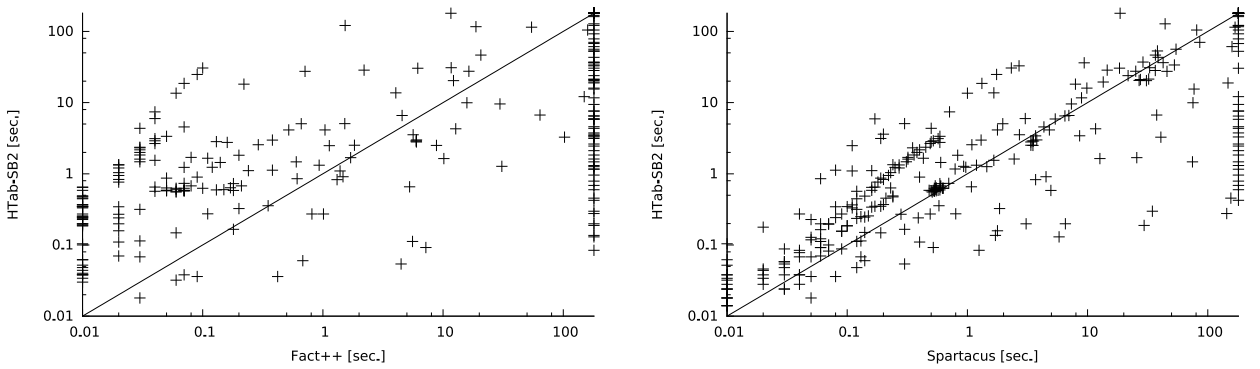
Table 6
Performance of $\texttt{HTab}_{\texttt{SB2}}$ in the LWB test set.

Class	#Inst	n^{100}_{HTab}	n^{100}_{SB2}	#ITrig	#TTrig	#Resch	T_{HTab}	T_{SB2}
k_branch_n	21	9	**21**	21	231	0	2224	**32**
k_branch_p	21	14	**21**	20	350	0	1428	**39**
k_grz_n	21	21	21	0	0	0	**0**	1
k_grz_p	21	21	21	1	1	0	**0**	**0**
k_path_n	21	21	21	21	252	0	**2**	3
k_path_p	21	21	21	20	230	0	**2**	3
k_ph_n	18	21	21	0	0	0	**93**	104
k_ph_p	21	7	7	0	0	0	**2517**	2535
k_poly_n	21	21	21	0	0	0	**13**	17
k_poly_p	21	21	21	0	0	0	**13**	17

Table 7
Results for $\texttt{HTab}_{\texttt{SB2}}$, Fact++ and Spartacus of the complete benchmark.

		$\texttt{HTab}_{\texttt{SB2}}$	Fact++	Spartacus
Solved	QBFLib	**168**	129	156
	LWB	**193**	164	173
	Total	**361**	293	329
Time	QBFLib	**3266**	8547	4712
	LWB	**2751**	7758	6539
	Total	**6017**	16 306	11 251
Sat	QBFLib	**64**	42	59
	LWB	**102**	87	91
	Total	**166**	129	150
Unsat	QBFLib	**104**	87	97
	LWB	**91**	77	82
	Total	**195**	164	179
TO	QBFLib	**4**	43	16
	LWB	**14**	43	34
	Total	**18**	86	50

behaves better than Fact++ and Spartacus, solving more formulas in less time. Among Fact++ and Spartacus, the latter is the one performing more similarly to $\texttt{HTab}_{\texttt{SB2}}$. However, the scatter plots of the runtimes of $\texttt{HTab}_{\texttt{SB2}}$ against Fact++ and Spartacus shown in Figs. 5a and 5b, respectively, reveal that there exists many instances in which both Fact++ and Spartacus outperform $\texttt{HTab}_{\texttt{SB2}}$. As a matter of fact, if we discard the timeouts and concentrate only on the total time required by instances solved by all three solvers, it turns out that Fact++ (638 seconds) outperforms $\texttt{HTab}_{\texttt{SB2}}$ (727 seconds) and Spartacus (847 seconds). $\texttt{HTab}_{\texttt{SB2}}$ consistently outperforms Fact++ and Spartacus in problems coming from the k_branch class in the LWB benchmarks, were we have already seen that the effects of symmetric blocking are more evident.



(a) $\texttt{HTab}_{\mathrm{SB2}}$ vs Fact++ (b) $\texttt{HTab}_{\mathrm{SB2}}$ vs Spartacus

Fig. 5. Running time comparison for $\texttt{HTab}_{\mathrm{SB2}}$, Fact++ and Spartacus.

6. Discussion

In this paper we exploited symmetries in a modal tableaux. We briefly presented an algorithm to detect symmetries for the basic modal logic introduced in [9]. Given a formula φ the algorithm generates a graph such that the automorphism group of the graph is isomorphic to the symmetry group of the formula. Layering is incorporated by duplicating the literal nodes occurring at different modal depth. Then we presented mechanisms that use symmetry information to block the triggering of the ($\Diamond$) rule in the tableaux. In particular, we introduced three different blocking conditions and show that they do not compromise completeness of the basic tableaux calculus. Finally we evaluated empirically these blocking conditions. Experimental results show that structured modal benchmarks are highly symmetric, that the applicability of the blocking mechanism highly depends on the problem class at hand, and that important performance gains can be obtained in some classes while imposing only a reasonable overhead.

Even though, in this article, we focused on the basic modal logic (with only one binary accessibility relation) it is clear that the approach we presented generalizes to the multi-modal, many-dimensional case with minimal changes. Layering symmetries, on the other hand, strongly depends on the possibility of working over trees (see [9] for further details). But even in those cases, global symmetries could still be used for symmetric blocking. We plan to investigate symmetric blocking conditions for other modal logics and also for related description logics as future research. It would be particularly interesting to test our approach on the large ontology repositories available in that community.

Our empirical results also clearly shows that symmetric blocking only exploits a small portion of the symmetry of the original formula, as many symmetric classes of problems do not trigger the symmetric blocking condition. Further testing is needed, and also other approaches to exploiting modal symmetries like, e.g., symmetry breaking predicates [3].

Appendix A

A.1. Tools and benchmarks

Available at: http://cs.famaf.unc.edu.ar/~ezequiel/sb-am.

A.2. Translations from QBF to basic modal logic

Let $\varphi = Q_1 p_1 \dots Q_m p_m \theta(p_1, \dots, p_m)$ be a prenex QBF formula. By $U(\varphi) = \langle Q_i \mid Q_i = \forall \text{ and } 1 \leq i \leq m \rangle$ we denote the sequence of $\forall$-*quantifiers* in the prefix of φ. By $E(\varphi) = \langle Q_i \mid Q_i = \exists \text{ and } 1 \leq i \leq m \rangle$ we denote the sequence of $\exists$-*quantifiers* in the prefix of φ. Given a sequence of quantifiers S, by S_i we denote the i-th quantifier in the sequence. S_1 denotes the first element of the sequence. By $|S|$ we denote the size of the sequence.

For each existential quantifier $\exists_i$, we define a *collapse level*, $CL(\exists_i)$, which corresponds to the index of the *last* $\forall$-quantifier occurring before $\exists_i$.

For $i, j \in \mathbb{N}, i \leq j$ we define $\Box^{(i)}\varphi = \varphi \wedge \Box\varphi \wedge \Box^2\varphi \wedge \ldots \wedge \Box^{i-1}\varphi \wedge \Box^i\varphi$, and, $\Box^{(i,j)}\varphi = \Box^i\varphi \wedge \Box^{i+1}\varphi \wedge \ldots \wedge \Box^j\varphi$. For $i, j, k \in \mathbb{N}, j \leq k$, we define $S(i, j, k)$ as

$$S(i, j, k) = \Box^{(j,k)}(\neg p_i \vee \Box p_i) \wedge \Box^{(j,k)}(p_i \vee \Box \neg p_i).$$

Definition 19 *(Collapse1 translation).* Let $\varphi = Q_1 p_1 \dots Q_m p_m\ \theta(p_1, \dots, p_m)$ be a CNF QBF formula and $m = |U(\varphi)|$. We define its translation to basic modal logic as the conjunction of the following formulas:

$$
\begin{array}{lll}
\text{(Root)}: & q_0, & \\
\text{(Level)}: & \bigwedge_{j\neq k} \Box^j(\neg q_j \vee \neg q_k) & 0 \leq j,k \leq m, \\
\text{(Peel\&Branch)}: & \bigwedge B(i,j) & 0 \leq j < m \text{ and } U(\varphi)_{(j+1)} = \forall_i, \\
\text{(Propagate-}\forall\text{)}: & \bigwedge S(i,j,m-1) & 1 \leq j \leq m-1 \text{ and } U(\varphi)_j = \forall_i, \\
\text{(Propagate-}\exists\text{)}: & \bigwedge S(i,j,m-1) & 1 \leq k \leq |E(\varphi)|, E(\varphi)_k = \exists_i \text{ and} \\
 & & j = CL(\exists_k), j \leq m, \\
\text{(Matrix)}: & \bigwedge \Box^m(\neg q_m \vee C_j) & \text{for all clauses } C_j \in \theta,
\end{array}
$$

where $B(i,j)$ is defined as

$$B(i,j) = \Box^j(\neg q_j \vee \neg\Box(\neg q_{j+1} \vee \neg p_i)) \wedge \Box^j(\neg q_j \vee \neg\Box(\neg q_{j+1} \vee p_i)).$$

Definition 20 *(Collapse2 translation).* Let $\varphi = Q_1 p_1 \ldots Q_m p_m\ \theta(p_1,\ldots,p_m)$ be a CNF QBF formula and $m = |U(\varphi)|$. We define its translation to basic modal logic as the conjunction of the following formulas:

$$
\begin{array}{lll}
\text{(Peel\&Branch)}: & \bigwedge B(i,j) & 0 \leq j < m \text{ and } U(\varphi)_j = \forall_i, \\
\text{(Propagate-}\forall\text{)}: & \bigwedge S(i,j,m-1) & 1 \leq j \leq m-1 \text{ and } U(\varphi)_j = \forall_i, \\
\text{(Propagate-}\exists\text{)}: & \bigwedge S(i,j,m-1) & 1 \leq k \leq |E(\varphi)|, E(\varphi)_k = \exists_i \text{ and} \\
 & & j = CL(\exists_k), j \leq m, \\
\text{(Matrix)}: & \bigwedge \Box^m C_j & \text{for all clauses } C_j \in \theta,
\end{array}
$$

where $B(i,j)$ is defined as

$$B(i,j) = \Box^j\neg\Box\neg p_i \wedge \Box^j\neg\Box p_i.$$

References

[1] B. Krishnamurthy, Short proofs for tricky formulas, Acta Inform. 22 (3) (1985) 253–275.
[2] J. Crawford, A theoretical analysis of reasoning by symmetry in first-order logic, in: Proceedings of AAAI'92 Workshop on Tractable Reasoning, San Jose, 1992, pp. 17–22.
[3] J. Crawford, M. Ginsberg, E. Luks, A. Roy, Symmetry-breaking predicates for search problems, in: Proceedings of KR'96, 1996, pp. 148–159.
[4] F. Aloul, A. Ramani, I. Markov, K. Sakallah, Solving difficult instances of boolean satisfiability in the presence of symmetry, EEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 22 (9) (2003) 1117–1137.
[5] B. Benhamou, L. Sais, Theoretical study of symmetries in propositional calculus and applications, in: Proceedings of CADE-11, 1992, pp. 281–294.
[6] B. Benhamou, L. Sais, Tractability through symmetries in propositional calculus, J. Automat. Reason. 12 (1) (1994) 89–102.
[7] G. Audemard, B. Mazure, L. Sais, Dealing with symmetries in quantified boolean formulas, in: Proceedings of SAT'04, 2004, pp. 257–262.
[8] D. Déharbe, P. Fontaine, S. Merz, B. Woltzenlogel Paleo, Exploiting symmetry in SMT problems, in: Proceedings of CADE-23, in: LNCS, Springer, 2011, pp. 222–236.
[9] C. Areces, E. Orbe, Symmetries in modal logics (Submitted for publication). Available at http://cs.famaf.unc.edu.ar/~ezequiel/sb-am.
[10] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, 2001.
[11] P. Blackburn, J. van Benthem, F. Wolter, Handbook of Modal Logic, Studies in Logic and Practical Reasoning, vol. 3, Elsevier Science Inc., New York, 2006.
[12] P. Patel-Schneider, R. Sebastiani, A new general method to generate random modal formulae for testing decision procedures, J. Artificial Intelligence Res. 18 (2003) 351–389.
[13] J. Fraleigh, V. Katz, A First Course in Abstract Algebra, Addison-Wesley, 2003.
[14] C. Areces, R. Gennari, J. Heguiabehere, M. de Rijke, Tree-based heuristics in modal theorem proving, in: Proceedings of ECAI'2000, Berlin, 2000, pp. 199–203.
[15] S. Fortin, The graph isomorphism problem, Tech. rep. 96-20, University of Alberta, Edomonton, Alberta, Canada, 1996.
[16] E. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, J. Comput. System Sci. 25 (1) (1982) 42–65.
[17] P. Darga, M. Liffiton, K. Sakallah, I. Markov, Exploiting structure in symmetry detection for CNF, in: Proceedings of DAC'04, 2004, pp. 530–534.
[18] T. Junttila, P. Kaski, Engineering an efficient canonical labeling tool for large and sparse graphs, in: Proceedings of ALENEX'07, SIAM, 2007.
[19] P. Balsiger, A. Heuerding, S. Schwendimann, A benchmark method for the propositional modal logics K, KT, S4, J. Automat. Reason. 24 (3) (2000) 297–317.
[20] G. Sutcliffe, The TPTP problem library and associated infrastructure: the FOF and CNF parts, v3.5.0, J. Automat. Reason. 43 (4) (2009) 337–362.
[21] C. Areces, J. Heguiabehere, hGen: a random CNF formula generator for hybrid languages, in: Proceedings of M4M-3, Nancy, France, 2003.
[22] E. Giunchiglia, M. Narizzano, A. Tacchella, Quantified boolean formulas satisfiability library (QBFLIB), http://www.qbflib.org, 2001.
[23] R. Ladner, The computational complexity of provability in systems of modal propositional logic, SIAM J. Comput. 6 (3) (1977) 467–480.
[24] R. Goré, Tableau methods for modal and temporal logics, in: M. D'Agostino, D. Gabbay, R. Haehnle, J. Posegga (Eds.), Handbook of Tableau Methods, Springer, Netherlands, 1999, pp. 297–396.
[25] I. Horrocks, U. Hustadt, U. Sattler, R. Schmidt, Computational modal logic, in: J.V.B. Patrick Blackburn, F. Wolter (Eds.), Handbook of Modal Logic, in: Studies in Logic and Practical Reasoning, vol. 3, Elsevier, 2007, pp. 181–245.
[26] D. Götzmann, M. Kaminski, G. Smolka, Spartacus: a tableau prover for hybrid logic, Electron. Notes Theor. Comput. Sci. 262 (2010) 127–139.
[27] C. Areces, E. Orbe, Dealing with symmetries in modal tableaux, in: D. Galmiche, D. Larchey-Wendling (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods, in: Lecture Notes in Computer Science, vol. 8123, Springer, Berlin, 2013, pp. 13–27.
[28] G. Hoffmann, C. Areces, Htab: a terminating tableaux system for hybrid logic, Electron. Notes Theor. Comput. Sci. 231 (2009) 3–19 (Proceedings of the 5th Workshop on Methods for Modalities, M4M5 2007).
[29] M. Kaminski, G. Smolka, Hybrid tableaux for the difference modality, Electron. Notes Theor. Comput. Sci. 231 (2009) 241–257.
[30] I. Horrocks, P.F. Patel-Schneider, Optimizing description logic subsumption, J. Logic Comput. 9 (3) (1999) 267–293.
[31] R. Zabih, D.A. McAllester, A rearrangement search strategy for determining propositional satisfiability, in: Proceedings of AAAI'88, vol. 88, 1988, pp. 155–160.

[32] D. Tsarkov, I. Horrocks, Fact++ description logic reasoner: system description, in: Proceedings of the Third International Joint Conference on Automated Reasoning, IJCAR'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 292–297.
[33] K. Schild, A correspondence theory for terminological logics: preliminary report, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence, vol. 1, IJCAI'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 466–471.