# Dealing with Symmetries in Modal Tableaux

Carlos Areces and Ezequiel Orbe[⋆]

FaMAF, U. Nacional de Córdoba (Argentina) and CONICET

**Abstract.** We present a technique that is able to detect symmetries in formulas of the basic modal logic ($\mathcal{BML}$). Then we introduce a modal tableaux calculus for $\mathcal{BML}$ with a blocking mechanism that takes advantage of symmetry information about the input formula to restrict the application of the ($\Diamond$) rule. We prove completeness of the calculus. Finally, we empirically evaluate both, the detection technique and the blocking mechanism in different modal benchmarks.

## 1 Introduction

In this work we investigate symmetries in modal logics. We discuss how to detect them in modal formulas and how to exploit them in a modal tableaux prover.

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) since the introduction in [21] of symmetry inference rules to strengthen resolution-based proof systems for propositional logic which lead to much shorter proofs of certain difficult problems (e.g., the pigeonhole problem). Since then many articles discuss how to detect and exploit symmetries. Most of them can be grouped into two different approaches: static symmetry breaking [11, 12, 1] and dynamic symmetry breaking [7, 8]. In the former, symmetries are detected and eliminated from the problem statement before the SAT solver is used, i.e., they work as a preprocessing step, whilst in the latter symmetries are detected and broken during the search space exploration. Despite their differences they share the same goal: to identify symmetric branches of the search space and guide the SAT solver away from symmetric branches already explored.

Research involving other logics has been done in the last years, e.g. [5, 14]. To the best of our knowledge, symmetries remain largely unexplored in automated theorem proving for modal logics. We investigated the theoretical foundations to exploit symmetries in a number of modal logics in [4]. In this paper we present a graph-based method to detect *layered symmetries* for the basic modal logic $\mathcal{BML}$ that extend the method presented in [4]. A layered symmetry is a more flexible notion of symmetry that can be defined in modal logics that enjoy the

[⋆] This work was partially supported by grants ANPCyT-PICT-2008-306, 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". We thank Guillaume Hoffmann for useful comments and help with `HTab`.

tree model property. Then we show how to use symmetry information in a modal tableaux calculus by presenting a novel blocking mechanism for a modal tableaux called *symmetry blocking*, that blocks the application of the $(\Diamond)$ rule if it has been already applied to a symmetric $\neg\Box$-formula. Finally we evaluate empirically both the detection technique and the blocking mechanism.

*Outline.* Section 2 introduces the required definitions on modal language and symmetries. Section 3 presents a graph construction algorithm to detect symmetries in modal formulas. Section 4 presents a $\mathcal{BML}$ tableaux calculus with symmetry blocking and proves it correctness. Finally, Section 5 presents experimental results about the detection construction and the effects of symmetry blocking in modal benchmarks. Section 6 concludes with some final remarks.

## 2 Definitions

In this section we introduce the basic notions concerning modal languages and permutations. In what follows, we will assume basic knowledge of classical modal logics and refer the reader to [9, 10] for technical details. Let $\mathsf{PROP} = \{p_1, p_2, \ldots\}$ be a countably infinite set of *propositional variables*. The set of (basic) modal formulas $\mathsf{FORM}$ is defined as

$$\mathsf{FORM} := p \mid \neg p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg\Box\varphi \mid \Box\varphi,$$

where $p \in \mathsf{PROP}$, $\varphi, \psi \in \mathsf{FORM}$. We will discuss only the mono-modal case, but the results we present extend naturally to the multi-modal case.

A *propositional literal* $l$ is either a propositional variable $p$ or its negation $\neg p$. The set of literals over $\mathsf{PROP}$ is $\mathsf{PLIT} = \mathsf{PROP} \cup \{\neg p_i \mid p_i \in \mathsf{PROP}\}$. A set of 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 $v \subseteq \mathsf{PROP}$ as $p \in v$ if $p \in L$ and $p \notin v$ if $\neg p \in L$. For $S \subseteq \mathsf{PROP}$, the *consistent and complete set of literals generated by* $S$ (notation $L_S$) is $S \cup \{\neg p \mid p \in \mathsf{PROP} \backslash S\}$.

A modal formula is in *modal conjunctive normal form (modal CNF)* if it is a conjunction of modal CNF clauses. A *modal CNF 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 modal CNF clause. Every modal formula can be transformed into an equisatisfiable formula in modal CNF in polynomial time [24]. In what follows assume that modal formulas are in modal CNF. Also, we often represent a modal CNF formula as a set of modal CNF clauses (interpreted conjunctively), and each modal CNF clause as a set of propositional and modal literals (interpreted disjunctively). The set representation disregards order and multiplicity of clauses and literals in a formula.

Models and semantics for modal CNF formulas are defined in the usual way. A (pointed) model is a tuple $\langle w, W, R, V\rangle$ such that $W$ 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 model. We define $\models$ for modal CNF formulas, clauses and literals as

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

The modal depth of a formula $\varphi$ in CNF (notation $\mathsf{md}(\varphi)$) is the maximum nesting of $\Box$ operators that occurs in $\varphi$.

A *permutation* is a bijective function $\rho : \mathsf{PLIT} \mapsto \mathsf{PLIT}$. We will define permutations using cyclic notation [16], e.g., $\rho = (p\ \neg q)(\neg p\ q)$ is the permutation that makes $\rho(p) = \neg q$, $\rho(\neg q) = p$, $\rho(\neg p) = q$ and $\rho(q) = \neg p$ 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. If $L$ is a set of literals then $\rho(L) = \{\rho(l) \mid l \in L\}$. Given a modal CNF formula $\varphi$ and a permutation $\rho$ we define $\rho(\varphi)$ as the formula obtained by simultaneously replacing all literals $l$ in $\varphi$ by $\rho(l)$. We say that a permutation $\rho$ is *consistent* if for every literal $l$, $\rho(\neg l) = \neg\rho(l)$. We say that a permutation $\rho$ is a *symmetry* for $\varphi$ if $\varphi = \rho(\varphi)$ when $\varphi$ is represented using sets. For example, consider the formula $\varphi = (\neg p \vee r) \wedge (q \vee r) \wedge \Box(\neg p \vee q)$, then the permutation $\rho = (p\ \neg q)(\neg p\ q)$ is a consistent symmetry of $\varphi$.

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 formula layers are semantically independent of each other (see [2] for details), i.e., at different layers the same propositional literal can be assigned a different value. This enables the definition of *layered permutations* [4], that let us define a different permutation at each modal depth.

A *finite permutation sequence* $\bar\rho$ is either the empty sequence $\bar\rho = \langle\rangle$ or $\bar\rho = \rho : \bar\rho_2$ with $\rho$ a permutation and $\bar\rho_2$ a permutation sequence. We sometimes use the notation $\bar\rho = \langle\rho_1, \ldots, \rho_n\rangle$ instead of $\bar\rho = \rho_1{:}\ldots{:}\rho_n{:}\langle\rangle$. Let $|\bar\rho| = n$ 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 $i \geq n$ and $\bar\rho_1 = \bar\rho$). $\bar\rho(i)$ is $\rho_i$ if $1 \leq i \leq |\bar\rho|$ and $\bar\rho(i) = \rho_{Id}$ otherwise, for $\rho_{Id}$ the identity permutation. A permutation sequence is *consistent* if all its permutations are consistent.

Let $\varphi$ be a modal CNF formula and $\bar\rho$ a permutation sequence. Then $\bar\rho(\varphi)$ is defined recursively as

$$\begin{array}{rll}
\langle\rangle(\varphi) = \varphi & & \\
(\rho_1{:}\bar\rho_2)(l) = \rho_1(l) & & \text{for } l \in \mathsf{PLIT} \\
(\rho_1{:}\bar\rho_2)(\Box C) = \Box\bar\rho_2(C) & & \\
\bar\rho(C) = \{\bar\rho(A) \mid A \in C\} & & \text{for } C \text{ a clause or a formula.}
\end{array}$$

One benefit of using a different permutation at each modal depth is that this enables symmetries (layered symmetries) to be found, that would not be found otherwise. 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 symmetries then $\varphi$ has none. However, the permutation sequence $\langle \rho_1, \rho_2 \rangle$ generated by $\rho_1 = (p\ \neg q)(\neg p\ q)$ and $\rho_2 = (p\ \neg r)(\neg p\ r)$ is a layered symmetry of $\varphi$.

**Definition 1 ($\bar{\rho}$-simulation [4]).** *Let $\bar{\rho}$ be a permutation sequence. 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 family of relations $Z_{\bar{\rho}_i} \subseteq W \times W'$, $1 \leq i$, that satisfies the following conditions:*

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

*We say that two 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 while $\mathcal{M} \underline{\rightarrow}_{\bar{\rho}} \mathcal{M}'$ implies $\mathcal{M}' \underline{\rightarrow}_{\bar{\rho}^{-1}} \mathcal{M}$, the relation $\underline{\rightarrow}_{\bar{\rho}}$ is not symmetric. From the definition of $\bar{\rho}$-simulations it intuitively follows that while they do not preserve validity of modal formulas (as is the case with bisimulations) they do preserve validity of *permutations* of formulas (see [4] for details).

**Proposition 1.** *Let $\bar{\rho}$ be a consistent layered permutation, $\varphi$ 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 $\mathcal{M}' \models \bar{\rho}(\varphi)$.*

## 3 Detecting Modal Symmetries

We present a graph-based technique for the detection of symmetries in modal CNF formulas. In propositional logic, it is standard to reduce the problem of finding symmetries in formulas to the problem of finding automorphisms in colored graphs where graphs are constructed from formulas in such a way that its automorphism group is isomorphic to the symmetry group of the formula [1, 11]

Although known to be an NP problem, it is not known if the graph automorphism problem is P or NP-complete [15]. Nevertheless there exist polynomial time algorithms for many special cases [23], and the are many efficient tools [13, 20] capable of compute a set of irredundant generators [16] for the automorphism group of very large graphs efficiently.

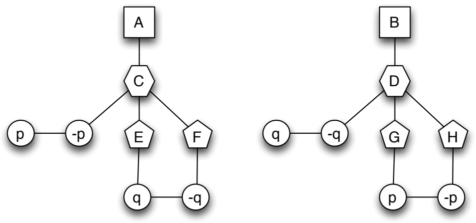
In [4] we extended the graph construction for propositional formulas presented in [1] to a wide range of modal logics. In this article we extend this construction to detect layered symmetries for $\mathcal{BML}$. In what follows, let $Var(\varphi, n)$ denote the set of variables occurring in $\varphi$ at modal depth $n$. Clauses occurring at modal depth 0 are called *top clauses*, and clauses occurring in modal literals are called *modal clauses.*

**Definition 2.** *Let $\varphi$ be a modal CNF formula of modal depth $n$. The colored graph $G(\varphi) = (V, E)$ corresponding to $\varphi$ is constructed as follows:*

1. *For each propositional variable $p \in Var(\varphi, i)$ with $0 \leq i \leq n$:*
   (a) *Add two* literal *nodes of color 0: one labeled $p_i$ and one labeled $\neg p_i$.*
   (b) *Add an edge between these two nodes to ensure Boolean consistency.*
2. *For each top clause $C$ of $\varphi$ add a* clause *node of color 1.*
3. *For each propositional literal $l$ occurring in $C$, add an edge from $C$ to the corresponding literal node $l_{\mathsf{md}(C)}$.*
4. *For each modal literal $\Box C'$ ($\neg\Box C'$) occurring in $C$:*
   (a) *Add a* clause *node of color 2 (color 3) to represent $C'$.*
   (b) *Add an edge from $C$ to this node.*
   (c) *Repeat the process from point 3 for each literal (propositional or modal) occurring in $C'$.*

This construction creates a graph with 4 colors and at most $2|V| \times (\mathsf{md}(\varphi) + 1) + \#TopClauses + \#ModalClauses$ nodes.

*Example 1.* Let us consider the following modal CNF formula $\varphi = \neg\Box(\neg p \vee \Box q \vee \Box\neg q) \wedge \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$. The associated 4-colored graph, $G(\varphi)$, is shown below (colors are represented by shapes in the figure).



$A = \neg\Box(\neg p \vee \Box q \vee \Box\neg q)$
$B = \neg\Box(\neg q \vee \Box p \vee \Box\neg p)$
$C = \neg p \vee \Box q \vee \Box\neg q$
$D = \neg q \vee \Box p \vee \Box\neg p$
$E = \Box q$
$F = \Box\neg q$
$G = \Box p$
$H = \Box\neg p$

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

Notice the way literals are handled during the construction of $G(\varphi)$: 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 colored differently. This avoids spurious permutations mapping $\Box C$ literals to $\neg\Box C$ literals and the other way around. The following Proposition is proved in [4].

**Proposition 2 ([4]).** *Let $\varphi$ be a modal CNF formula and $G(\varphi) = (V, E)$ the colored graph obtained following the construction of Definition 2. Then every symmetry $\bar{\rho}$ of $\varphi$ corresponds one-to-one to an automorphism $\pi$ of $G(\varphi)$.*

## 4 Symmetry Blocking

We present a labeled tableaux calculus [18] for the basic modal logic $\mathcal{BML}$ and a dynamic blocking mechanism that uses symmetry information from the input formula. We will use standard tableaux prefixed rules and notation which we briefly review for completeness. Let $\mathsf{PREF}$ be an infinite, non-empty set of *prefixes*. Here we will set $\mathsf{PREF} = \mathbb{N}$. Given $\varphi$ 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 $\sigma$. 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 $\sigma'$ is accessible via $R$ from $\sigma$.

A tableaux for $\varphi \in \mathcal{BML}$ 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 $\varphi$ is a modal CNF formula and $C$ a modal clause.

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

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

$^1$ $\sim C$ is the CNF of the negation of $C$. The prefix $\sigma'$ 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. A branch is *closed* if it contains both $\sigma{:}p$ and $\sigma{:}\neg p$, and it is *open* otherwise. A branch is *saturated* if no rule can be further applied in the branch (notice that the conditions we imposed on rule application lead to saturation after a finite number of steps).

Let $\mathsf{Tab}(\varphi)$ be the set of tableaux for $\varphi$ 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 $\mathcal{BML}$.

**Theorem 1.** *For any $\varphi \in \mathcal{BML}$, any $\mathsf{T} \in \mathsf{Tab}(\varphi)$ is finite. Moreover $\mathsf{T}$ has a saturated open branch if and only if $\varphi$ is satisfiable.*

This completes our brief introduction of the standard tableaux calculus for $\mathcal{BML}$. We are interested in 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 a symmetric formula before. Notice that this restriction cannot affect termination or soundness of the calculus.

Let $\mathsf{T} \in \mathsf{Tab}(\varphi)$, and let $\Theta$ be a branch of $\mathsf{T}$. For $\sigma \in \mathsf{PREF}$ we will use $\mathsf{depth}(\sigma)$ for the distance from the root prefix (in particular, if $\sigma$ is the prefix of $\varphi$ then $\mathsf{depth}(\sigma) = 0$). Given a prefixed formula $\sigma{:}\varphi$ and a permutation sequence $\bar{\rho}$, we define $\bar{\rho}(\sigma{:}\varphi) = \sigma{:}\bar{\rho}_{\mathsf{depth}(\sigma)+1}(\varphi)$. Finally, given a modal formula $\varphi$, we define $Var(\varphi)$ as the set of propositional variables occurring in $\varphi$; for $S$ a set of modal formulas, let $Var(S) = \bigcup_{\varphi \in S} Var(\varphi)$.

**Symmetry Blocking:** Let $\bar{\rho}$ be a layered symmetry of $\varphi$, and let $\Theta$ be a branch in a tableau of $\varphi$. The rule $(\Diamond)$ cannot be applied to $\sigma{:}\bar{\rho}(\neg\Box\psi)$ on $\Theta$ if it has been applied to $\sigma{:}\neg\Box\psi$ and $Var(\bar{\rho}(\neg\Box\psi)) \cap Var(\Gamma(\sigma)) = \emptyset$, for $\Gamma(\sigma) = \{\psi \mid \sigma : \Box\psi \in \Theta\}$ the set of $\Box$-formulas occurring at prefix $\sigma$[1].

Note that symmetry blocking is a dynamic condition: after being blocked, a $\neg\Box$-formula can be scheduled for expansion 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. From now on, we consider that branches in $\mathsf{Tab}(\varphi)$ are saturated using symmetry blocking.

## 4.1 Completeness

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

**Definition 3.** *Given an open saturated branch $\Theta$ of the tableaux $\mathsf{T} \in \mathsf{Tab}(\varphi)$, we define a model $\mathcal{M}^{\Theta} = \langle W^{\Theta}, R^{\Theta}, V^{\Theta}\rangle$ as:*

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

Notice that $\mathcal{M}^{\Theta}$ is always a tree. Given a tree and a permutation sequence, we can construct a new model as follows.

**Definition 4 ($\bar{\rho}$-*image* of a tree model).** *Given a pointed tree model $\mathcal{M} = \langle w, W, R, V\rangle$ and a permutation sequence $\bar{\rho}$. Let $\mathsf{depth}(v)$ be the distance of $v$*

[1] A more strict symmetry blocking condition is possible, where instead of requiring $Var(\bar{\rho}(\neg\Box\psi)) \cap Var(\Gamma(\sigma)) = \emptyset$ we verify that the variables at each modal depth of $\bar{\rho}(\neg\Box\psi)$ are disjoint from those in $\Gamma(\sigma)$ (i.e., $\forall n.\, Var(\bar{\rho}(\neg\Box\psi), n) \cap Var(\Gamma(\sigma), n) = \emptyset$). But this more aggressive blocking condition did not have an impact in our experiments. It is easy to find cases where a $\neg\Box$-formula is only blocked under the more strict condition, but we did not find any such case in the test sets we investigated.

*to the root $w$ (in particular $\mathsf{depth}(w) = 0$). Define $\mathcal{M}_{\bar{\rho}} = \langle w, W_{\bar{\rho}}, R_{\bar{\rho}}, V_{\bar{\rho}}\rangle$ (the* $\bar{\rho}$-image *of $\mathcal{M}$) as the model identical to $\mathcal{M}$ except that*

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

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

Given $\Theta$ a saturated open branch, let $\Sigma$ be the set of prefixes added to $\Theta$ by the application of the ($\Diamond$) rule to a $\neg\Box$-formula that has a symmetric $\neg\Box$-formula blocked by symmetry blocking. Intuitively, $\Sigma$ contains the roots of the sub-models that need a symmetric counterpart in the completion of $\mathcal{M}^{\Theta}$.

Let $M[\Sigma] = \{\mathcal{M}^{\Theta}[\sigma] \mid \sigma \in \Sigma\}$. This set contains the sub-models to which we need to construct a symmetric sub-model. By $M[\Sigma]_{\bar{\rho}} = \{\mathcal{M}^{\Theta}[\sigma]_{\bar{\rho}} \mid \mathcal{M}^{\Theta}[\sigma] \in M[\Sigma]\}$ we denote the set of $\bar{\rho}$-images corresponding to the set of sub-models $M[\Sigma]$. Intuitively, $M[\Sigma]_{\bar{\rho}}$ is the set of models that we need to "glue" to the model $\mathcal{M}^{\Theta}$ to obtain a complete model.

**Definition 5 (Symmetric Extension).** *Given a saturated open branch $\Theta$, a model $\mathcal{M}^{\Theta} = \langle W^{\Theta}, R^{\Theta}, V^{\Theta}\rangle$ and a set of symmetric pointed sub-models $M[\Sigma]_{\bar{\rho}}$. The symmetric extension of $\mathcal{M}^{\Theta}$ is the model $\mathcal{M}^{\Theta}_{\bar{\rho}} = \langle W^{\Theta}_{\bar{\rho}}, R^{\Theta}_{\bar{\rho}}, V^{\Theta}_{\bar{\rho}}\rangle$ where:*

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

*for all $\langle \tau_{\sigma'}, W, R, V\rangle \in M[\Sigma]_{\bar{\rho}}$, where $\tau_{\sigma'}$ is the element corresponding to $\sigma'$ in the disjoint union.*

Note that we are *gluing* the symmetric sub-models to the original model by adding an edge from the element $\sigma$ to the root, $\tau_{\sigma'}$, of the symmetric sub-model if there is an edge from $\sigma$ to the root, $\sigma'$, of the original sub-model. To be sure that this construction is sound we have to check that after adding these sub-models to the original model, the $\Box$-formulas holding at $\sigma$, $\Gamma(\sigma)$, still hold.

The following lemma is the key to prove completeness of our calculus. First recall the following well known result.

**Proposition 3.** *Let $\varphi$ be a modal formula and $\mathsf{PROP}$ a set of propositional variables such that $Var(\varphi) \subseteq \mathsf{PROP}$. Let $\mathcal{M} = \langle W, R, V\rangle$ be a model such that $V : W \mapsto \mathcal{P}(\mathsf{PROP})$. Then $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}{\upharpoonright}Var(\varphi), w \models \varphi$.*

**Lemma 1.** *Let $\varphi$ be a modal CNF formula, $\bar{\rho}$ a symmetry of $\varphi$ and $\Theta$ be a saturated open branch of $\mathsf{T} \in \mathsf{Tab}(\varphi)$. Let $\sigma$ 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. Then $\bar{\rho}(\psi) \bigwedge \Gamma(\sigma)$ is satisfiable.*

*Proof.* Given that $\Theta$ is a saturated open branch, we know that $\sigma R \sigma' \in \Theta$ and that $\sigma' : \psi \bigwedge \Gamma(\sigma) \in \Theta$. From $\Theta$ 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, $\mathcal{M}^{\Theta}, \sigma' \models \psi \bigwedge \Gamma(\sigma)$ with $(\sigma, \sigma') \in R^{\Theta}$.

Let $\mathcal{M}^{\Theta}[\sigma'] = \langle \sigma', W', R', V' \rangle$ be the sub-model rooted at $\sigma'$ with $W' = W^{\Theta}[\sigma']$, $R' = R^{\Theta} \upharpoonright W^{\Theta}[\sigma']$ and $V' = V^{\Theta} \upharpoonright W^{\Theta}[\sigma']$. Then $\mathcal{M}^{\Theta}[\sigma'] \models \psi \bigwedge \Gamma(\sigma)$. Let $\mathcal{N} = \mathcal{M}^{\Theta}[\sigma'] \upharpoonright Var(\psi) = \langle \sigma', W', R', V'_{\mathcal{N}} \rangle$ and $\mathcal{R} = \mathcal{M}^{\Theta}[\sigma'] \upharpoonright Var(\Gamma(\sigma)) = \langle \sigma', W', R', V'_{\mathcal{R}} \rangle$. By Proposition 3 $\mathcal{N} \models \psi$ and $\mathcal{R} \models \bigwedge \Gamma(\sigma)$.

Let $\mathcal{N}' = \bar{\rho}(\mathcal{N}) = \langle \sigma', W', R', V''_{\mathcal{N}} \rangle$. By construction, $\mathcal{N} \underline{\rightarrow}_{\bar{\rho}} \mathcal{N}'$ and therefore $\mathcal{N}' \models \bar{\rho}(\varphi)$. Finally, let $\mathcal{U} = \mathcal{N}' \cup \mathcal{R} = \langle \sigma, W', R', V''' \rangle$ where $V'''(w) = V''_{\mathcal{N}}(w) \cup V'_{\mathcal{R}}(w)$ for all $w \in W'$. By the symmetry blocking condition we know that $Var(\bar{\rho}(\psi)) \cup Var(\Gamma(\sigma)) = \emptyset$ and therefore $L_{V''_{\mathcal{N}}(w)} \cap L_{V'_{\mathcal{R}}(w)} = \emptyset$ for all $w \in W'$. It follows that no contradiction will arise when doing $V''_{\mathcal{N}}(w) \cup V'_{\mathcal{R}}(w)$ and hence that the valuation function $V'''(w)$ is well defined.

Now we have to prove that $\mathcal{U} \models \bar{\rho}(\psi) \bigwedge \Gamma(\sigma)$. First we prove that $\mathcal{U} \models \bar{\rho}(\psi)$. Take the restriction of $\mathcal{U}$ to $Var(\bar{\rho}(\psi))$, $\mathcal{U} \upharpoonright Var(\bar{\rho}(\psi))$. By construction of $\mathcal{U}$, we know that $\mathcal{U} \upharpoonright Var(\bar{\rho}(\psi)) = \mathcal{N}'$ and that $\mathcal{N}' \models \bar{\rho}(\psi)$. By Proposition 3, $\mathcal{U} \models \bar{\rho}(\psi)$. That $\mathcal{U} \models \bigwedge \Gamma(\sigma)$ holds, follows by the same argument using the model $\mathcal{R}$.

We are now ready to prove a correspondence between formulas in the branch $\Theta$ and truth in the symmetric extension of model built from it.

**Lemma 2.** *Let $\Theta$ be a saturated open branch of a tableau $\mathsf{T} \in \mathsf{Tab}(\varphi)$ and $\bar{\rho}$ a symmetry of $\varphi$. 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 the syntactic structure of $\psi$.

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

$[\psi = \neg p]$ Since $\Theta$ 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 $(\Diamond)$, $\sigma R \sigma'$, $\sigma'{:}\theta \in \Theta$. By definition of $R^{\Theta}_{\bar{\rho}}$ and induction hypothesis: $(\sigma, \sigma') \in R^{\Theta}_{\bar{\rho}}$ and $\mathcal{M}^{\Theta}_{\bar{\rho}}, \sigma' \models \theta$. 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 symmetry blocking. In this case, $\neg\Box\theta = \bar{\rho}(\neg\Box\chi) = \neg\Box\bar{\rho}(\chi)$. By saturation of $(\Diamond)$ we have that $\sigma R \sigma'$, $\sigma'{:}\chi \in \Theta$. Moreover, we have that $(\sigma, \sigma') \in R^{\Theta}$ and that $\mathcal{M}^{\Theta}, \sigma' \models \chi$. By definition of the symmetric extension of $\mathcal{M}^{\Theta}$ we have that $(\sigma, \tau_{\sigma'}) \in R^{\Theta}_{\bar{\rho}}$ and $\mathcal{M}^{\Theta}_{\bar{\rho}}, \tau_{\sigma'} \models \bar{\rho}(\chi)$. 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 $\sigma'$ such that $(\sigma, \sigma') \in R^{\Theta}_{\bar{\rho}}$ then this holds trivially. Otherwise, let $\sigma'$ 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 not a 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 Lemma 1, 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 2.** *The tableau calculus with symmetry blocking for the modal logic $\mathcal{BML}$ is sound and complete.*

*Proof.* The tableaux calculus we introduced is clearly sound for $\mathcal{BML}$ and symmetry blocking cannot affect soundness. For completeness, let $\Theta$ be an open saturated branch of the tableau $\mathsf{T} \in \mathsf{Tab}(\varphi)$. Since $0 : \varphi \in \Theta$, by Lemma 2 we get that $\varphi$ is satisfiable.

## 5 Experimental Evaluation

In this section, we empirically evaluate the detection of symmetries and the effect of symmetry blocking in modal benchmarks.

For testing we use `HTab` [19], a tableaux prover developed in Haskell, for the hybrid logic $\mathcal{H}(:, \mathsf{E}, \mathsf{D}, \Diamond^-, \downarrow)$ with reflexive, transitive and symmetric modalities[2]. `HTab` includes a series of optimizations that are enabled by default, namely, semantic branching, dependency-directed backtracking, lazy branching, unit propagation and eager unit propagation. For symmetry detection we use Bliss [20], a graph automorphism tool. Bliss takes as input a graph specification and returns a set of generators for the automorphism group of the graph. For each formula the computed symmetries are saved in file. A formula together with its symmetry file are given as input to `HTab`. All tests are run on an Intel Core i7 2.93GHz with 16GB of RAM.

### 5.1 Symmetry Detection

In Section 3 we presented a graph construction to compute symmetries of modal formulas. It remains to test how often symmetries appear in modal benchmark and how hard it is to actually find them. Our testbed is made of 1134 structured instances: 378 instances from the Logics Workbench Benchmark for $\mathcal{BML}$ (LWB_K) [6] (distributed in 9 problem classes)[3] and 756 instances from the QBFLib Benchmarks [17] (distributed in 12 problem classes). Instances from the QBFLib benchmarks were translated to $\mathcal{BML}$ using a variation of Ladner's translation [22] that reduces the modal depth of the resulting formula. For each instance we generate the corresponding graph and feed it to Bliss.

Table 1 shows the results for the LWB_K and QBFLib benchmarks. Columns #Inst and #Sym are the number of instances and the number of instances with at least one symmetry, respectively. Column $T$ is the total time, in seconds, needed to process all instances. The table clearly shows that both benchmarks are highly symmetric. Formulas in QBFLib are large (ranging into the 250 megabytes after translation into $\mathcal{BML}$) which explains the difference in the time required to process all instances.

[2] Download page: http://www.glyc.dc.uba.ar/intohylo/htab.php

[3] We use negate the formulas in LWB_K before constructing the tableaux. *_p classes are unsatisfiable, and *_n classes are satisfiable.

The LWB_K benchmark presents a behavior that coincide with our expectations: the existence of symmetries is driven by the codification used in each problem class. Table 2 shows detailed results for this benchmark. Column AvGen is the average number of generators. It shows that some problem classes (`k_branch`, `k_path`, `k_grz`, `k_ph` and `k_poly`) exhibit a large amount of symmetries while others exhibit none (`k_d4`, `k_dum`, `k_t4p`) or very few symmetries (`k_lin`). The QBFLib benchmark also exhibit a large amount of symmetries: 11 of 12 problem classes have a 100% of highly symmetric instances, although in this case, the translation from QBF to $\mathcal{BML}$ accounts for a large number of the detected symmetries (see [5] for details on symmetry detection for QBF formulas). With respect to efficiency, Table 1 shows that for the LWB_K benchmark the time required to compute the symmetries is negligible. For the QBFLib benchmark, it greatly varies depending on the problem class and is directly correlated to the size of the instances.

**Table 1.** Detected Symmetries

| | #Inst | #Sym | $T$ |
|---|---|---|---|
| LWB_K | 378 | 208 | 10.2 |
| QBFLib | 756 | 746 | 16656 |

**Table 2.** Symmetries in LWB_K

| Class | #Inst | #Sym | AvGen |
|---|---|---|---|
| `k_branch` | 42 | 42 | 12 |
| `k_d4` | 42 | 0 | 0 |
| `k_dum` | 42 | 0 | 0 |
| `k_grz` | 42 | 42 | 4 |
| `k_lin` | 42 | 1 | 1 |
| `k_path` | 42 | 42 | 35 |
| `k_ph` | 42 | 39 | 1 |
| `k_poly` | 42 | 42 | 18 |
| `k_t4p` | 42 | 0 | 0 |

We also tested symmetry detection on a random testbed. The testbed contains 19000 formulas in CNF generated using hGen [3]. We fix the maximum modal depth of the formulas ($D$) to 3. Instances are distributed in 10 sets. For each set we fix the number of propositional variables ($N$) (from 10 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 value of $L/N$ are likely to be satisfiable, whilst formulas with greater values of $L/N$ are often unsatisfiable. Each set contain 100 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 the generation of a larger number of clauses). As expected, the number of symmetries diminish with the addition of new, random clauses.

Figure 1 shows the percentage of symmetric instances for each value of the ratio $L/N$. The figure shows that 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. Again this coincides with expectations: large values of $L/N$ results from a high number of clauses in the instances, reducing the possibility of symmetries.

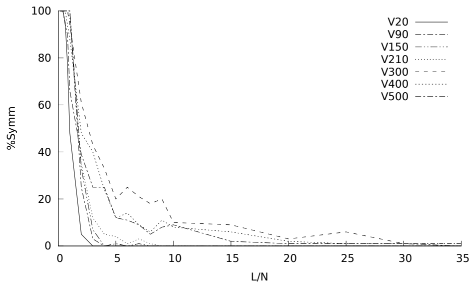


**Fig. 1.** % of random symmetric instances.

## 5.2 Symmetry Blocking

The implementation of symmetry blocking (SB) in HTab is straightforward: whenever there is a $\neg\Box$-formula scheduled for expansion, the solver checks if there is a symmetric formula already expanded. If this is the case, it blocks the $\neg\Box$-formula 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.

Our testbed includes 954 symmetric instances from the LWB_K and QBFLib benchmarks (for time constraints and space limitations, we do not report results on random formulas). Table 3 presents the results with and without symmetry blocking (HTab+SB and HTab, respectively). Columns $T_1$ and $T_2$ are total times, including symmetry computation, in seconds, on the complete testbed, including and excluding timeouts, respectively (timeout was set to 600 seconds). It shows that HTab+SB outperforms HTab: HTab+SB requires less time to solve all the instances and solves 7 instances more than HTab (HTab+SB is able to solve 9 instances that timeout with HTab, but timeouts in other 2 that HTab is able to solve). It also shows that there is a large number of formulas with timeouts (mostly from QBFLib's formulas)[4].

**Table 3.** Total Times with SB

| Solver | #Suc | #TO | $T_1$ | $T_2$ |
|---|---|---|---|---|
| HTab+SB | 318 | 636 | 9657 | 391167 |
| HTab | 311 | 643 | 10634 | 396434 |

Figure 2 presents a scatter plot of the running times for the 320 formulas that succeed in at least one of the configurations. The $x$ axis gives the running times of HTab without symmetry blocking, whereas the $y$ axis gives the running times of HTab+SB. 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. Approximately half of the instances report a performance gain while the other half report a slight performance degradation. Degradation is due to the extra overhead imposed by the blocking mechanism

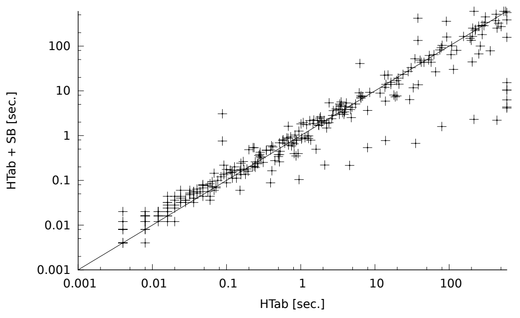


**Fig. 2.** Performance of HTab vs. HTab with symmetry blocking on all formulas.

[4] Large formulas from QBFLib often resulted in timeouts. Time constraints put restrictions on the timeout value we could use. We are currently running further tests with larger timeouts.

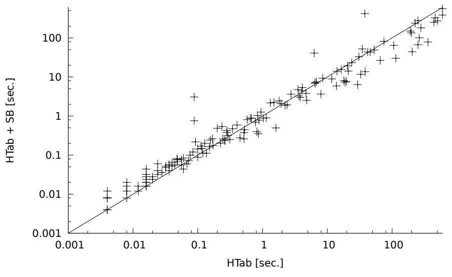
on instances that never trigger SB. Nevertheless, degradation is negligible for most of the instances.

Table 4 shows information about the application of SB. Column #Trig is the number of instances that trigger SB at least once, columns $B_1$ and $B_2$ are the number of times that SB is triggered and the number of times that SB is not correct, respectively.

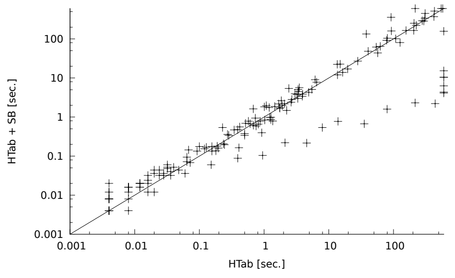
**Table 4.** Applications of SB.

| Status | #Inst | #Trig | $B_1$ | $B_2$ |
|---|---|---|---|---|
| Satisfiable | 157 | 73 | 6319 | 6278 |
| Unsatisfiable | 163 | 79 | 1038 | 87 |

For satisfiable instances, SB triggers many times but in most cases the blocking condition fails later in the branch (remember that the blocking condition is dynamic). Figure 3a) shows that there is still an improvement for several instances (44% for all instances, 59% for instances that triggered SB). It also shows that degradation is almost negligible. This tells us that even in cases where SB is not correct, delaying the processing of symmetric formulas is beneficial because the branch has more information available that can avoid branching or close the branch more rapidly.



a) Satisfiable formulas



b) Unsatisfiable formulas

**Fig. 3.** Performance of `HTab` vs. `HTab`+SB.

For unsatisfiable instances, SB triggers less often than for satisfiable instances, but most of the blockings are correct (but notice that SB is not validated if the branch closes). Figure 3b) shows a great improvement for several instances. Also `HTab`+SB proves 7 more instances than `HTab`. Degradation is 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.

Table 5 presents detailed results for the LWB_K benchmark. We show the number of the largest formula that could be solved within a time limit of 100 ($n_{100}$) and a time limit of 600 ($n_{600}$), and the total time required to process all the class, including timeouts ($T$). It shows that the effectiveness of SB is highly dependent on the problem class. For some classes SB provide an impor-

**Table 5.** Effect of SB on the LWB_K

| Class | HTab+SB | | | HTab | | |
|---|---|---|---|---|---|---|
| | $n_{100}$ | $n_{600}$ | $T$ | $n_{100}$ | $n_{600}$ | $T$ |
| `k_branch_p` | 21 | 21 | 59.760 | 13 | 15 | 4402.130 |
| `k_branch_n` | 9 | 10 | 7010.200 | 8 | 10 | 7197.000 |
| `k_grz_p` | 21 | 21 | 0.508 | 21 | 21 | 0.276 |
| `k_grz_n` | 21 | 21 | 0.632 | 21 | 21 | 0.380 |
| `k_path_p` | 21 | 21 | 4.542 | 21 | 21 | 3.812 |
| `k_path_n` | 21 | 21 | 5.348 | 21 | 21 | 3.792 |
| `k_ph_p` | 7 | 8 | 8116.900 | 7 | 8 | 8095.48 |
| `k_ph_n` | 21 | 21 | 177.560 | 21 | 21 | 178.579 |
| `k_poly_p` | 21 | 21 | 29.068 | 21 | 21 | 22.949 |
| `k_poly_n` | 21 | 21 | 29.534 | 21 | 21 | 24.229 |

tant performance gain (e.g., `k_branch`). On the other hand, for several highly symmetric classes, SB does not make a difference as it never gets triggered. In other words, symmetric blocking only addresses a small part of the symmetries usually present in modal formulas.

## 6 Conclusions

In this paper we exploited symmetries in a modal tableuax. First we presented a method to detect symmetries for $\mathcal{BML}$ that extend the method presented in [4] to detect layered symmetries. Given a formula $\varphi$ it 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 a blocking mechanism for a modal tableaux that uses symmetry information only. This mechanism, called *symmetry blocking*, blocks the application of the ($\Diamond$) rule if it has been already applied to a symmetric $\neg\Box$-formula. Finally we evaluated empirically both, the detection technique and the blocking mechanism. Experimental results shows that structured modal benchmarks are highly symmetric and that our detection algorithm is efficient at computing symmetries. In the case of symmetry blocking, results shows 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 on problem classes not suited for this blocking mechanism. Further testing is needed, and also other approaches to exploiting modal symmetries like, e.g., symmetry breaking predicates [12].

## References

1. F. Aloul, A. Ramani, I. Markov, and K. Sakallah. Solving difficult instances of Boolean satisfiability in the presence of symmetry. *IEEE Trans. of CAD*, 22(9):1117–1137, 2003.

2. C. Areces, R. Gennari, J. Heguiabehere, and M. de Rijke. Tree-based heuristics in modal theorem proving. In *Proc. of ECAI'2000*, pages 199–203, Berlin, 2000.
3. C. Areces and J. Heguiabehere. hGen: A Random CNF Formula Generator for Hybrid Languages. In *Proc. of M4M-3*, Nancy, France, 2003.
4. C. Areces, G. Hoffmann, and E. Orbe. Symmetries in modal logics. In D. Kesner and P. Viana, editors, *Proc. of LSFA'12*, pages 27–44, 2013.
5. G. Audemard, B. Mazure, and L. Sais. Dealing with symmetries in quantified boolean formulas. In *Proc. of SAT'04*, pages 257–262, 2004.
6. P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. of Aut. Reas.*, 24(3):297–317, 2000.
7. B. Benhamou and L. Sais. Theoretical study of symmetries in propositional calculus and applications. In *Proc. of CADE-11*, pages 281–294, 1992.
8. B. Benhamou and L. Sais. Tractability through symmetries in propositional calculus. *J. of Aut. Reas.*, 12(1):89–102, 1994.
9. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.
10. P. Blackburn, J. van Benthem, and F. Wolter. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier Science Inc., New York, 2006.
11. J. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proc. of AAAI'92 Work. on Tractable Reasoning*, pages 17–22, San Jose, 1992.
12. J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proc. of KR'96*, pages 148–159, 1996.
13. P. Darga, M. Liffiton, K. Sakallah, and I. Markov. Exploiting structure in symmetry detection for CNF. In *Proc. of DAC'04*, pages 530–534, 2004.
14. D. Déharbe, P. Fontaine, S. Merz, and B. Woltzenlogel Paleo. Exploiting symmetry in SMT problems. In *Proc. of CADE-23*, LNCS, pages 222–236. Springer, 2011.
15. S. Fortin. The graph isomorphism problem. Technical report, Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada, 1996.
16. J. Fraleigh and V. Katz. *A first course in abstract algebra*. Addison-Wesley, 2003.
17. E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `http://www.qbflib.org`.
18. R. Goré. Tableau methods for modal and temporal logics. In M. DAgostino, D. Gabbay, R. Hhnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Springer Netherlands, 1999.
19. G. Hoffmann and C. Areces. Htab: A terminating tableaux system for hybrid logic. In *Proc. of M4M-5*, 2007.
20. T. Junttila and P. Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. of ALENEX'07*. SIAM, 2007.
21. B. Krishnamurthy. Short proofs for tricky formulas. *Act. Inf.*, 22(3):253–275, 1985.
22. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. on Comp.*, 6(3):467–480, 1977.
23. E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42 – 65, 1982.
24. P. Patel-Schneider and R. Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *J. of Art. Int. Res.*, 18:351–389, 2003.