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 $S \cup \{\neg p \mid p \in \mathsf{PROP} \backslash S\}$.

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