Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 262 (2010) 127–139
www.elsevier.com/locate/entcs

Spartacus: A Tableau Prover for Hybrid Logic

Daniel Götzmann[1] Mark Kaminski[1] Gert Smolka[1]

Saarland University
Saarbrücken, Germany

Abstract

Spartacus is a tableau prover for hybrid multimodal logic with global modalities and reflexive and transitive relations. Spartacus is the first system to use pattern-based blocking for termination. To achieve a competitive performance, Spartacus implements a number of optimization techniques, including a new technique that we call lazy branching. We evaluate the practical impact of pattern-based blocking and lazy branching for the basic modal logic **K** and observe high effectiveness of both techniques.

Keywords: hybrid logic, modal logic, tableau algorithms, decision procedures, automated reasoning

1 Introduction

Automated reasoning in modal and description logics (DL) is an active field of research. Arguably the most successful approach to modal reasoning are tableau-based methods. Several of the most prominent DL reasoners, including FaCT++ [31] and RacerPro [14], are based on tableau algorithms. In the presence of global modalities or transitive relations, the naive tableau construction strategy, sufficient in the case of basic modal logic, no longer terminates. To regain termination, one employs blocking [22]. Most of the established blocking techniques are derived from Kripke's chain-based approach [24]. Kaminski and Smolka [21,22] propose a different blocking technique, called pattern-based blocking. They conjecture that pattern-based blocking may display a better performance than the established techniques. Our goal is to show that pattern-based blocking is useful even for **K**, where blocking is not required for termination.

Spartacus is a tableau prover for hybrid multimodal logic with global modalities. It supports reasoning in the presence of reflexive and transitive relations. In contrast to other systems, Spartacus uses pattern-based blocking to achieve termination. Similarly to FaCT++ [30,31,32], Spartacus schedules pending rule appli-

[1] Email: {goetzmann,kaminski,smolka}@ps.uni-sb.de

doi:10.1016/j.entcs.2010.04.010

cations using a configurable priority queue, which allows for a fine-grained control over the rule application strategy. To achieve a reasonable performance on realistic inputs, Spartacus implements a number of optimizations, including term simplification (also called "normalization" [17]), Boolean constraint propagation, semantic branching and backjumping [32]. Moreover, Spartacus implements a new technique, called *lazy branching*. Lazy branching is a generalization of lazy unfolding [32], an effective optimization technique from DL reasoning.

Spartacus is written in Standard ML and compiled with MLton. The source code and test data are available from www.ps.uni-sb.de/theses/goetzmann/. A detailed description of Spartacus can be found in [12].

We evaluate the effects of pattern-based blocking and lazy branching, and compare the performance of Spartacus with that of other reasoners for modal and description logics. Both techniques prove highly effective.

2 The Logic

Spartacus decides the satisfiability problem for $\mathcal{H}(E, @)$, the basic hybrid logic extended with global modalities. Notationally, our description of $\mathcal{H}(E, @)$ follows [21]. We distinguish between variables for *states* (x, y), *properties* (p, q), and *relations* (r). From these variables, the *expressions* of $\mathcal{H}(E, @)$ can be obtained by the following grammar:

$$s, t \ ::= \ \dot{\top} \mid p \mid \dot{x} \mid \dot{\neg} s \mid s \mathbin{\dot{\wedge}} s \mid \langle r \rangle s \mid Es \mid @xs$$

We employ the usual abbreviations $s \mathbin{\dot{\vee}} t := \dot{\neg}(\dot{\neg} s \mathbin{\dot{\wedge}} \dot{\neg} t)$, $[r]s := \dot{\neg}\langle r \rangle \dot{\neg} s$, and $As := \dot{\neg} E \dot{\neg} s$. For details on $\mathcal{H}(E, @)$ and related logics, see [1].

In addition to expressions of the above form, Spartacus accepts reflexivity and transitivity assertions of the form *Reflexive* r and *Transitive* r.

Except for the details of the blocking mechanism, the calculus underlying Spartacus is a restriction of the system in [22] to $\mathcal{H}(E, @)$. The calculus works on formulas of the form sx where s is a negation normal expression of $\mathcal{H}(E, @)$ and x a state. The use of the state variable x in a formula sx corresponds to the use of *prefixes* [6] or *nodes* [19] in related calculi. Since for later discussion the treatment of equality in Spartacus is inessential, let us consider the following restriction of the calculus to **K**.

$$\mathcal{R}_{\dot{\neg}} \ \frac{px,\ (\dot{\neg} p)x}{\bot} \qquad \mathcal{R}_{\dot{\wedge}} \ \frac{(s \mathbin{\dot{\wedge}} t)x}{sx,\ tx} \qquad \mathcal{R}_{\dot{\vee}} \ \frac{(s \mathbin{\dot{\vee}} t)x}{sx \ \mid \ tx} \qquad \mathcal{R}_{\Diamond} \ \frac{(\langle r \rangle s)x}{rxy,\ sy}\ y \text{ fresh} \qquad \mathcal{R}_{\Box} \ \frac{([r]s)x,\ rxy}{sy}$$

The symbol $\bot$ marks closed branches. The formula rxy specifies that y has to be *accessible* from x, corresponding to the notation $x \Diamond_r y$ in [6] and $\langle x, y \rangle \in \mathcal{E}_A(r)$ in [19].

3 Pattern-Based Blocking

Pattern-based blocking (PBB) in Spartacus is implemented following [21]. The technique yields termination in the presence of nominals, transitive relations, global