KSP: A resolution-based prover for multimodal K

Cláudia Nalon[1] Ullrich Hustadt[2] Clare Dixon[2]

[1] Department of Computer Science, University of Brasília, Brasília, Brazil
nalon@unb.br
[2] Department of Computer Science, University of Liverpool, Liverpool, UK
{U.Hustadt, CLDixon}@liverpool.ac.uk

Abstract: We briefly present a new normal form and calculus for the propositional basic multimodal logic K_n. We introduce the prover KSP that implements that calculus and methods for normal form transformation. We present an experimental evaluation that compares KSP with a range of existing reasoners for K_n.

1 Introduction

Automatic theorem proving for the basic multimodal logic K_n, the logic that extends propositional logic with unary operators $[a]$ and $\langle a\rangle$ for each index a in a finite set A, has been intensely studied as it is able to express non-trivial problems in Artificial Intelligence and other areas. For instance, the description logic ALC, which has been applied to terminological representation, is a syntactic variant of K_n. Problems in Quantified Boolean Propositional Logic can also be translated into K_n. In this paper we briefly describe a new normal form and calculus for K_n, the prover KSP implementing the calculus, and present an experimental evaluation of KSP.

2 A Normal Form and Calculus for K_n

In [6] we have presented a novel resolution-based calculus for K_n. The calculus takes advantage of the following well-known properties of basic modal logic: (i) if a modal formula φ is satisfiable, then it is satisfiable in a Kripke model where the union of the accessibility relations is a tree; and (ii) in tree models, checking the local satisfiability of φ can be reduced to checking the local satisfiability of its subformulae at the depth of a model with corresponds to the modal level where those subformulae occur in φ.

The calculus operates on labelled clauses of the form $ml : C$ (literal clause), $ml : l \to [a]l'$ (positive a-clause), $ml : l \to \langle a\rangle l'$ (negative a-clause) where $ml \in \mathbb{N} \cup \{*\}$, C is a propositional clause, and l, l' are propositional literals. Intuitively, the label ml indicates the depth of the Kripke model at which the clause is true. The special label $*$ is used if a clause is true at all depths/worlds in a model and it normally only occurs in the normal form if we want to check a formula for global satisfiability instead of local satisfiability. Any formula of K_n can be transformed into an equi-satisfiable set of labelled clauses.

The calculus uses a simple form of unification. We define a partial function σ on sets of labels as follows: $\sigma(\{ml, *\}) = ml$; and $\sigma(\{ml\}) = ml$; otherwise, σ is undefined. The rules of the calculus, shown in Table 1, can only be applied if the unification of the labels involved is defined.

$$[\mathsf{LRES}]\ \frac{ml_1 : C \vee l \qquad ml_2 : D \vee \neg l}{ml : C \vee D} \qquad [\mathsf{MRES}]\ \frac{ml_1 : \ l_1 \to [a]l \qquad ml_2 : \ l_2 \to \langle a\rangle \neg l}{ml : \neg l_1 \vee \neg l_2}$$

$$[\mathsf{GEN1}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a]\neg l_1 \\ & \vdots \\ ml_m : & l'_m \to [a]\neg l_m \\ ml_{m+1} : & l' \to \langle a\rangle \neg l \\ ml_{m+2} : & l_1 \vee \ldots \vee l_m \vee l \end{array}}{ml : \ \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'}$$

$$[\mathsf{GEN2}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a]l_1 \\ ml_2 : & l'_2 \to [a]\neg l_1 \\ ml_3 : & l'_3 \to \langle a\rangle l_2 \end{array}}{ml : \ \neg l'_1 \vee \neg l'_2 \vee \neg l'_3}$$

$$[\mathsf{GEN3}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a]\neg l_1 \\ & \vdots \\ ml_m : & l'_m \to [a]\neg l_m \\ ml_{m+1} : & l' \to \langle a\rangle l \\ ml_{m+2} : & l_1 \vee \ldots \vee l_m \end{array}}{ml : \ \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'}$$

Table 1: Inference rules, where $ml = \sigma(\{ml_1, ml_2\})$ in LRES, MRES; $ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\})$ in GEN1, GEN3; and $ml = \sigma(\{ml_1, ml_2, ml_3\})$ in GEN2.

3 KSP

KSP is an implementation, written in C, of the calculus in Table 1. It also incorporates a range of methods to transform modal formulae into the clausal normal form that the calculus operates on. The main loop is based on the given-clause algorithm implementation in Otter, a variation of the set of support strategy.

Besides the basic calculus KSP uses three more inference rules including unit resolution. The user can restrict LRES by choosing ordered (clauses can only be resolved on their maximal literals with respect to an ordering chosen by the prover in such a way to preserve completeness), negative (one of the premises is a negative clause, i.e. a clause where all literals are negative), positive (one of the premises is a positive clause), or negative + ordered resolution (both negative and ordered resolution inferences are performed). The completeness of some of these refinements depends on the

particular normal form chosen. For a comprehensive description of KSP see [4], the prover itself is available at [5].

4 Evaluation

We have compared KSP with BDDTab, FaCT++ 1.6.3, InKreSAT 1.0, Spartacus 1.0, and a combination of the optimised functional translation [2] with Vampire 3.0 (OFT + Vampire).

Our benchmarks, available at [5], consist of three collections of modal formulae:

1. Modalised random QBF (MQBF) formulae [3] (1016 formulae, 617 known to be satisfiable and 399 known to be unsatisfiable).
2. LWB basic modal logic benchmark formulae, further subdivided into 18 classes [1] (1008 formulae, half are satisfiable and half are unsatisfiable by construction of the benchmark classes).
3. Randomly generated 3CNF_K formulae [7] over 3 to 10 propositional symbols with modal depth 1 or 2 (1000 formulae, 457 are known to be satisfiable and 464 are known to be unsatisfiable).

Figure 1 shows the performance of the various provers on these three collections of benchmark formulae, in particular, the graphs show how many formulae a prover can solve if given a certain amount of CPU time to solve each. KSP performs significantly better than any of the other provers on the MQBF collection. On the LWB collection overall, KSP performs about as well as BDDTab, FaCT++ and InKreSAT, while Spartacus performs best. A more detailed analysis shows that BDDTab and InKreSAT are the best performing provers on one LWB class each, OFT + Vampire on two, KSP on six, and Spartacus on eight classes. Finally, on the 3CNF_K collection, InKreSAT is the best performing prover and KSP the worst performing one.

In general, KSP performs best on formulae with high modal depth where atomic subformulae are (evenly) spread over a wide range of modal levels. The benchmarks indicate that KSP is a useful addition to the collection of provers that are already available for K_n.

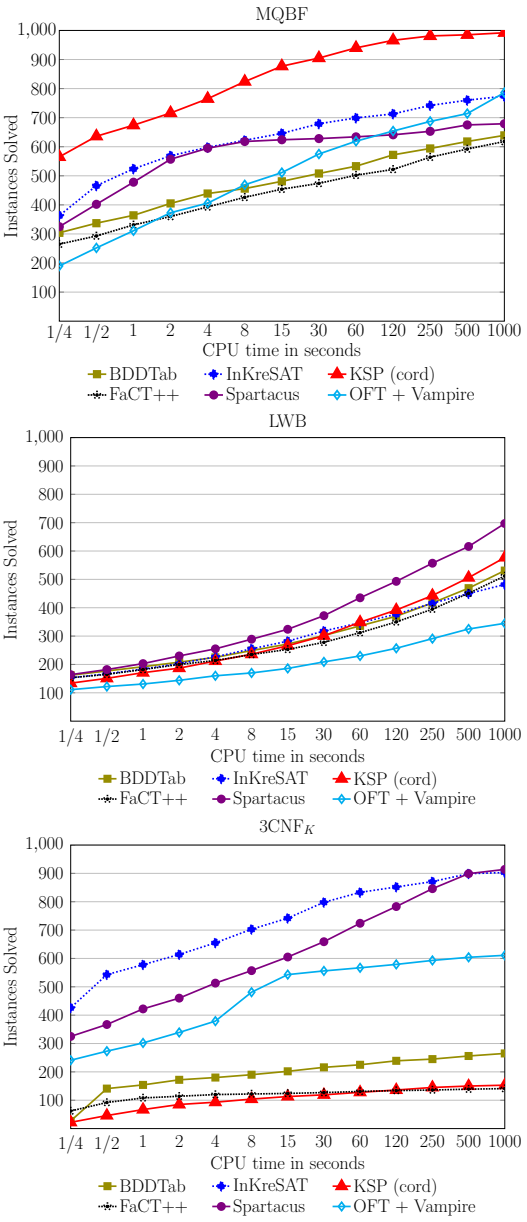

Figure 1: Benchmarking results

References

[1] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. Autom. Reasoning*, 24(3):297–317, 2000.

[2] Ian R. Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate Schmidt. Computational modal logic. In *Handbook of Modal Logic*, pages 181–245. Elsevier, 2006.

[3] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-2000 non-classical (modal) systems comparison. In *Proc. TABLEAUX 2000*, volume 1847 of *LNCS*, pages 52–56. Springer, 2000.

[4] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. KSP: A resolution-based prover for multimodal K. To appear in Proc. IJCAR 2016.

[5] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. KSP: sources and benchmarks. `http://www.cic.unb.br/~nalon/#software`.

[6] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. A modal-layered resolution calculus for K. In *Proc. TABLEAUX 2015*, volume 9323 of *LNCS*, pages 185–200. Springer, 2015.

[7] Peter F. Patel-Schneider and Roberto Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *J. Artif. Intell. Res. (JAIR)*, 18:351–389, 2003.