Kꓘ: a theorem prover for K

Andrei Voronkov

Computer Science Department, University of Manchester

1 General description

Nonclassical propositional logics play an increasing role in computer science. They are used in model checking, verification, and knowledge representation. Traditional decision procedures for these logics are based on semantic tableaux [6, 7, 1], SAT-based methods [4], or translation into classical logic [10]. In this system abstract we overview the system Kꓘ that implements the tableau method and the less traditional inverse method for propositional modal logic K.

The main purpose of Kꓘ is to compare top-down (tableau) and bottom-up (inverse) decision methods for K and other logics. It is expected to extend Kꓘ with several modal logics, description logics and quantified boolean formulas.

The theoretical basis for Kꓘ is presented in [13]. The system is implemented in ML and runs under Windows32 and Unix. No graphical interface is yet available.

Propositional K is PSPACE-complete. Tableau-based decision procedures for K can easily be implemented in polynomial space. It is not clear whether the inverse method can, in general, be implemented in polynomial space[1].

2 Preliminaries

For simplicity, we restrict ourselves to formulas built up from literals (i.e. propositional variables or their negations) using $\wedge$, $\vee$, $\Box$ and $\Diamond$. We denote propositional variables by lower-case letters, and formulas by upper-case letters. A *sequent* is a finite multiset of formulas. We denote sequents by Greek letters Γ and Δ. For a sequent $\Gamma = A_1, \ldots, A_n$, we denote by $\Diamond\Gamma$ the sequent $\Diamond A_1, \ldots, \Diamond A_n$. A sequent $A_1, \ldots, A_n$ is called *satisfiable* if it is true in some Kripke model and *unsatisfiable* otherwise.

3 The tableau method implementation

The tableau method implemented in Kꓘ is based on the sequent calculus Seq given in Figure 1. The sequent calculus is complete for unsatisfiability: a formula A is unsatisfiable if and only if it has a proof in Seq.

1 Recently, Hustadt and Schmidt [10] have shown that semantic tableaux can be polynomially simulated by a variant of resolution, but we do not have enough details to comment on the relevance of their results for the inverse method.

$$\text{axioms: } \Gamma, p, \neg p \qquad \frac{\Gamma, A, B}{\Gamma, A \wedge B}\ (\wedge)$$

$$\frac{\Gamma, A \quad \Gamma, B}{\Gamma, A \vee B}\ (\vee) \qquad \frac{\Delta, A}{\Gamma, \Box\Delta, \Diamond A}\ (\Diamond)$$

Fig. 1. Sequent calculus Seq

The proof-search by the tableau method is implemented in a straightforward way as a backward proof procedure for Seq. We use an efficient encoding of sequents and the heuristics selecting the shortest subformula in a sequent. We also tried to add to Seq a restricted but sound version of simplification rules

$$\frac{\Gamma[p \leftarrow \text{true}]}{\Gamma, p} \qquad \frac{\Gamma[p \leftarrow \text{false}]}{\Gamma, \neg p}$$

but it did not improve the performance.

4 The inverse method implementation

The inverse method is based on a bottom-up version of Seq. The sequent calculus for the inverse method depends on the goal formula G, is denoted by Inv_G and shown in Figure 2.

$$\text{axioms: } p, \neg p \qquad \frac{\Gamma, A, A}{\Gamma, A}\ (C)$$

$$\frac{\Gamma, A}{\Gamma, A \wedge B}\ (\wedge_l) \qquad \frac{\Gamma, B}{\Gamma, A \wedge B}\ (\wedge_r)$$

$$\frac{\Gamma, A \quad \Delta, B}{\Gamma, \Delta, A \vee B}\ (\vee)$$

$$\frac{\Gamma, A}{\Box\Gamma, \Diamond A}\ (\Diamond) \qquad \frac{\Gamma}{\Box\Gamma, \Diamond A}\ (\Diamond')$$

All formulas occurring in the rules are subformulas of G.

Fig. 2. Sequent calculus Inv_G

This calculus is complete in the following sense: a formula G is unsatisfiable if and only if it is derivable in Inv_G (completeness for sequents can also be proved but is formulated in a different way). The implementation of the inverse method is based on a standard saturation procedure:

1. Let S be a set of sequents, initially $\{p, \neg p \mid p, \neg p \text{ are subformulas of } G\}$.

2. Repeatedly apply all possible inference rules to sequents in S, adding their conclusions to S until no new sequents can be obtained or a derivation of G is found.

However, such a procedure is hopelessly inefficient compared to the tableau method, unless augmented with powerful redundancy criteria allowing us to get rid of redundant sequents and redundant inferences.

Several such redundancies are discussed in [13]. The redundancies used in our implementation are obtained by analyzing possible derivations in Seq and specializing Inv_G to avoid derivations that either cannot occur in Seq or correspond to redundant derivations in Seq. We also used subsumption as a redundancy criterion.

5 Experiments

In this section we consider experimental results obtained by running our system on 20 randomly generated benchmarks. The benchmarks were generated using the random formula generator implemented by Fabio Massacci[2] and discussed in [8].

Since our aim was a prototype implementation and the comparison of the inverse method with the tableau method, our current implementation is made in ML. A more efficient implementation in C++ will be released soon.

The results (in seconds, rounded down to an integer) are given in Table 1. The second column shows the results for the tableau method, the third column for the inverse method. The fourth column describes the naive combination of the two methods: two methods are run in parallel using equal system resources until at least one of them terminates. The bottom rows of the table give the average time for all 20 problems, the average time for 80% of the problems (excluding two fastest and two slowest) and the number of problems solved within some time intervals (for example, 1 minute). The best results are grayed.

We would like to emphasize that we compare a *very straightforward* implementation of the inverse method with a very straightforward implementation of the tableau method. Some remarks about further possible optimizations are included in Section 6.

As in the case of CASC competition for classical logic, one can note that the correlation between runtimes for different methods is quite loose. We do not want to discuss the big difference in performance on some benchmarks (though there is an easy explanation), for example problem 11 is solved in less than a second by the tableau method and in about 7 hours by the inverse method. Our implementations of both methods are not optimized which causes instability of experimental results.

[2] `http://hermes.dis.uniroma1.it/~massacci/TANCS/problem.html`. The generator was called with the -K option.

problem	tableau	inverse	mixed
1	1	4	2
2	403	1	2
3	1,296	913	1,826
4	12,642	1,264	2,528
5	10,323	1,135	2,270
6	4,383	4	8
7	2,120	65	130
8	16,332	8,477	16,954
9	102,339	4,187	8,374
10	7	23	14
11	0	23,198	0
12	3,690	17,512	7,380
13	15	586	30
14	910	21	42
15	275	11	22
16	31	789	62
17	23,290	1,370	2,740
18	11,626	46	92
19	701	20	40
20	1,348	72	144
avg	9,586	2,984	2,576
80%	4,131	1,186	1,083
10 sec	3	3	4
1 min	5	8	9
5 min	6	10	13
30 min	11	16	13

Table 1. Experimental results

6 Further developments

To implement the inverse method more efficiently, more investment in datastructures and algorithms in needed. ML is not very efficient (both in terms of time and space) for such datastructures. We are now implementing a C++ version of the prover. Many important optimizations have not been included in the system yet.

When we implemented the first version of KЯ, we were not aware of the optimizations used in the fastest tableau-based implementations and described in [7]. All redundancy criteria used to optimize our implementation are specific for the inverse method and cannot be used for tableaux. We are currently investigating which optimizations described in [7] can be applied to the inverse method. Our guess is that any optimizations not destroying the subformula property can be used in some form without losing completeness. Some optimizations of [7] do not preserve the subformula property, but in a very liberal way. For example, in the semantic splitting rule

$$\frac{\Gamma, A \quad \Gamma, \neg A, B}{\Gamma, A \vee B}\ (\vee)$$

the formula $\neg A$ may not be a subformula of the goal formula. However, it is not difficult to modify the inverse method by allowing negations of all subformulas to be used.

Acknowledgments

Part of this research was done in Uppsala University, where the author was supported by TFR grants. We thank Larisa Maksimova for helpful comments on modal logic.

References

1. F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In H.Boley and M.M.Richter, editors, *Processing Declarative Knowledge (Proceedings of the International Workshop PDK'91*, volume 567 of *Lecture Notes in Artificial Intelligence*, pages 67–86. Springer Verlag, 1991.
2. L. Catach. Tableaux: a general theorem prover for modal logics. *Journal of Automated Reasoning*, 7(4):489–510, 1991.
3. M. D'Agostino. Are tableaux an improvement of truth tables? *Journal of Logic, Language and Information*, 1:235–252, 1992.
4. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures: Case study of modal K. In M.A. McRobbie and J.K. Slaney, editors, *CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 583–597, 1996.
5. L.F. Goble. Gentzen systems for modal logic. *Notre Dame J. of Formal Logic*, 15:455–461, 1974.
6. I. Horrocks and P.F. Patel-Schneider. FaCT and DLP. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98*, volume 1397 of *Lecture Notes in Computer Science*, pages 27–30, Oisterwijk, The Netherlands, May 1998. Springer Verlag.
7. I. Horrocks and P.F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 1999. To appear.
8. U. Hustadt and R. Schmidt. On evaluating decision procedures for modal logic. In *IJCAI-97*, volume 1, pages 202–207, 1997.
9. U. Hustadt and R.A. Schmidt. Simplification and backjumping in modal tableau. In H. de Swart, editor, *TABLEAUX'98*, volume 1397 of *Lecture Notes in Computer Science*, pages 187–201, 1998.
10. U. Hustadt and R.A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In *IJCAI-99*, 1999. To appear.
11. S.Yu. Maslov. An inverse method for establishing deducibility of nonprenex formulas of the predicate calculus. In J.Siekmann and G.Wrightson, editors, *Automation of Reasoning (Classical papers on Computational Logic)*, volume 2, pages 48–54. Springer Verlag, 1983.
12. A. Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 648–662, 1992.
13. A. Voronkov. A bottom-up decision procedure for propositional K: theory and implementation. 1999. To be submitted.