Implementation of a tableau theorem prover for modal logic

Zhen Li, the University of Manchester.
Email: liz@cs.man.ac.uk

Introduction. *Modal logics* are well known as a formal way of handling notions of knowledge, believe and time. Many efforts have been taken to develop a deduction system for modal logics [7, 8, 1, 2]. Tableau system have been implemented in many theorem provers for modal logic or its variant, description logic [3, 6]. In this paper, we discuss the implementation issues of a modal logic theorem prover. In particular, we focus on tableau systems for modal logics, namely K and extensions of K with the axiom T, D, B, 4, and 5. Details about the implementation of the tableau algorithm for modal logic are described based on a system called MLTP (Modal Logic Tableaux Prover). Optimization techniques have been incorporated into MLTP. Effects of these techniques are analyzed using performance data from empirical tests.

Modal logics. Modal logics extend propositional logic with the necessity operator ($\Box$) and the possibility operator ($\Diamond$). The semantics of modal logics are defined by the usual Kripke semantics. Different modal logics are obtained from modal logic K by addition of any combination of extra axioms. Well-known axioms include the axioms T, 4, 5, B, and D. Each of them has a FOL correspondence property which defines a class of frames in which the axiom is valid. Table 1 gives the axioms and their correspondence properties.

Axiom	Property	FOL definition
T	reflexivity	$\forall x.xRx$
D	seriality	$\forall x \exists y.xRy$
B	symmetry	$\forall x, y.xRy \to yRx$
4	transitivity	$\forall x, y, z.(xRy \wedge yRz) \to xRz$
5	euclideanness	$\forall x, y, z.(xRy \wedge xRz) \to yRz$

Table 1: Axioms and their correspondence properties

Tableau Algorithm. Tableau systems are widely used as deductive systems to verify the validity of a formula in modal logics [4, 5]. Based on the Kripke model, labelled tableau systems have been developed to test the satisfiability in modal logics [4]. A labelled tableau system consists of labelled formulae. A labelled formula has the form $w : \phi$ where w is viewed as a world and ϕ is a modal formula. It has the semantic meaning that the formula ϕ is true in the world w. A labelled tableau is constructed by applying expansion rules. The expansion rules of MLTP are classified similarly as in [4]: α denotes conjunction formulae, β denotes disjunction formulae, ν denotes necessity formulae and π possibility formulae. The expansion rules are listed in Table 2, where w'' in the π-rule is new on the branch. These rules handle the four kinds of formulae (α, β, ν and π) in different ways in accordance with their semantics. To deal with the extra modal axioms, we use the *structural rules* shown in Table 3. In

α-rule	β-rule	ν-rule	π-rule
$\dfrac{w : \alpha}{\begin{array}{c} w : \alpha_1 \\ w : \alpha_2 \end{array}}$	$\dfrac{w : \beta}{w : \beta_1 \mid w : \beta_2}$	$\dfrac{w : \nu \ wRw'}{w' : \nu_0}$	$\dfrac{w : \pi}{\begin{array}{c} wRw'' \\ w'' : \pi_0 \end{array}}$

Table 2: Tableau expansion rules

T-rule	4-rule	5-rule	B-rule	D-rule
$\dfrac{}{wRw}$	$\dfrac{\begin{array}{c} wRw' \\ w'Rw'' \end{array}}{wRw''}$	$\dfrac{\begin{array}{c} wRw' \\ wRw'' \end{array}}{w'Rw''}$	$\dfrac{wRw'}{w'Rw}$	$\dfrac{}{wRw'}$

Table 3: Structural rules

Table 1, we showed that there is a relational property corresponding to each of the well-known axioms (T, D, 4, 5 and B). Essentially the corresponding structural rules complete the partial frame constructed thusfar in the derivation so that the frame satisfies the relational correspondence properties. For instance, to apply the structural rule of euclideanness we need to go through the frame, if there are two worlds w, w' with a common predecessor then a link between w and w' is added.

Implementation. As a test-bed for the tableau systems of modal logics described above MLTP is able to solve satisfiability problems of the modal logic K and the extensions with the axioms T, D, B, 4, 5 or their combinations. To improve the deduction performance MLTP also utilizes some optimization techniques.

System description. MLTP is implemented in the C programming language. C outperforms other languages like Prolog, LISP and Java in efficiency. As far as we know there is no other automated modal logic prover implemented in C (FACT is implemented in LISP, DLP in ML and Lotrec in Java). Trials with a C prover are therefore novel and interesting.
MLTP consists of five modules: the parser, the preprocessor, the inference engine and the output module. The parser takes a plain text file as input. The input file should include information about the user, the axioms, the properties of the variables, and the conjecture formula to be tested. The parser sets up the internal data structures holding this information. Based on these data structures, the preprocessor does preprocessing, which might include transformation into a normal form and simplification. In the inference engine, expansion rules and structural rules are repeatedly applied until the deduction terminates. Loop checking is performed to stop infinite derivations. Finally the derivations performed during the search are extracted and output in understandable format.

Data structures. In MTLP a formula is stored in a tree structure. Hence MTLP can deal with extremely long formulae as long as memory resources permit it. The tree structure also makes it easy to extract the subformulae, which is the most common operation during the derivation.
To have flexibility, the expansion rules are formalized into a general data structure which contains the information of the numerator and the denominator of the rules. In addition, the user can specify the order in which rules are applied. This information is stored in an array.

The basic data structure for a tableau branch in MLTP is a FIFO queue. The labelled formulae and the worlds and links of the frame are stored in a data structure named *inference node*. An inference node also holds the information about the premises and the rules that were used to generate the node. A FILO stack is used to implement backtracking over disjunctive branches.

Inference engine. The inference engine employs a depth-first left-to-right search strategy. Expansion rules are used one by one in the specified order and applied repeatedly to inference nodes in the derivation queue. Structural rules are applied at the beginning (for T and D) and after the applications of the π-rule (for 4, 5 and B). Clash checking is applied after each inference step. Loop checking is performed before each application of the π-rule. It prevents a potentially infinite branch by detecting two identical worlds. It is worth noting that when dealing with modal logics which could cause infinite branches, the π-rule should be applied after the other expansion rules, otherwise the prover could judge a branch as a loop by mistake (e.g. $\Box\Diamond p \wedge \Box(\neg p \wedge q)$ in $KT4$).

Optimization. Many techniques and strategies have been proposed to improve the efficiency of inference for modal logic. The optimization techniques implemented in MLTP are *unit propagation*, *simplification* and *backjumping*. Simplification is applied during preprocessing, unit propagation and backjumping are performed during the derivation.

Unit propagation, also called boolean constraint propagation, is a technique used to minimize nondeterministic expansion steps (branching). Branching significantly increases the search space. The algorithm performs clash detection before branching and prunes the branch that is found to be inconsistent.

Simplification aims to minimize the size of the problem formulae and therefore reduce the size of the search space. For this purpose, a set of simplification rules are applied to the formulae. Most of the rules correspond to the tautologies of propositional logic.

Backjumping, also known as dependency-directed backtracking, is used to avoid obvious inefficiency during the derivation. The derivation for a complex problem usually needs to go through a number of branches. It often happens that the closures of several branches are caused by the same unsatisfiability. Backjumping works by detecting this situation after expanding one of the branches and bypassing the others.

Evaluation and Conclusion. Empirical tests have been done to evaluate the performance of the MLTP system and the effectiveness of the optimization techniques. The results of the tests performed on randomly generated problems are plotted in Figure 1. They have shown that the three optimization techniques all contribute to the improvement of the derivation performance. The impact of unit propagation and simplification is greater on simple problems while backjumping improves the performance considerably on complex problems. It can be proved that our implementation is fair, therefore no inference of a formula could be postponed forever.

Our plans are to extend MLTP to reason about actions and plans of agents. We also intend to study and incorporate more optimization techniques and efficient data structures to speed up the derivation.

a)

b)

c)

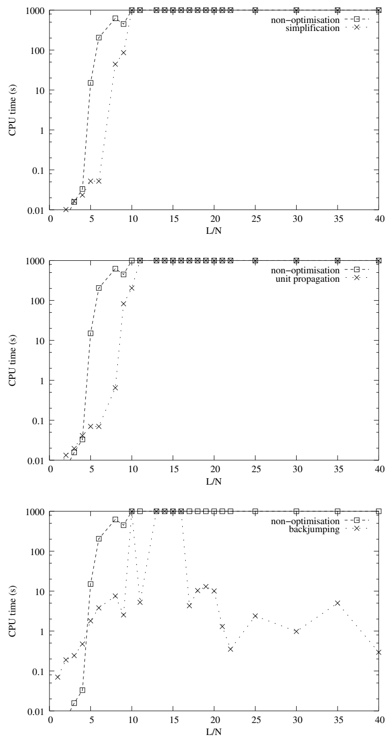

Figure 1: 50th percentile CPU time of tests with (a) simplification, (b) unit propagation and (c) backjumping.

References

[1] M. Baldoni. *Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension*. PhD thesis, Univ. degli Studi di Torino, Italy, 1998.

[2] Marcos A. Castilho, Luis Farias del Cerro, Olivier Gasquet, and Andreas Herzig. Modal tableaux with propagation rules and structural rules. *Fundamenta Informaticae*, 32(3-4):281–297, 1997.

[3] L. F. Cerro, D. Fauthoux, O. Gasquet, F. Massacci, A. Herzig, and D. Longin. Lotrec: Generic tableau prover for modal and description logic. *IJCAR'01, LNAI*, 2083:453–458, 2001.

[4] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Co., 1983.

[5] R. Goré. Tableau methods for modal and temporal logics. In M. D. Agostino, D. M. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of tableau methods*, pages 297–396. Kluwer, 1999.

[6] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[7] U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. *TABLEAUX 2000. LNAI*, 1847:67–71, 2000.

[8] Fabio Massacci. Single step tableaux for modal logics. *Journal of Automated Reasoning*, 24:319–364, 2000.