System description of $Tab_{FV}{}^{Prover}$: A theorem prover for first order modal logics

Virginie Thion
Université de Paris-Sud, L.R.I.
E-mail : Virginie.Thion@lri.fr

November 2001

Abstract

In previous papers [CMC00, CMC01], M. Cialdea Mayer and S. Cerrito have proposed a proof procedure for some variants of first order modal logics, called Tab_{FV}; the proposed calculi are free variable tableaux methods.

Essentially, a first order modal interpretation is a set of classical interpretations equipped with a binary relation, the accessibility relation [Kri63]. However, several choices can be made with respect to semantics, concerning the object domains, the designation of terms, the existence of objects [CMC00, CMC01].

So, several variants of quantified modal logics (QMLs) are possible, just by choosing different combinations of the cases considered above. We call them DDE variants (Domains/Designation/Existence variants) of QML. Mayer and Cerrito's systems are modular w.r.t. "DDE variants".

We have implemented $Tab_{FV}{}^{Prover}$, so as to obtain a theorem prover for fist-order modal logics . This paper aims to describe $Tab_{FV}{}^{Prover}$.

1 Introduction

The proof systems Tab_{FV} are calculi in the free-variable tableau style ([Fit88]), treating different DDE variants of first order modal logics. The notion of DDE variants is described in the papers [CMC00, CMC01] and we invite the reader to consult these papers for more details. We will recall it briefly here.

1.1 DDE variant

Different hypotheses can be considered on first order modal semantics, concerning the object domains, the designation of terms and the existence of objects :

The object domains (relation between the domains of the worlds) :

- constant domain : All worlds have the same domain. (This case is not treated by $Tab_{FV}{}^{Prover}$)
- cumulative domains : If the world w' is accessible from the world w then the domain of w is included in the domain of w'.
- variable domains : Nothing is supposed concerning the domains of worlds.

The designation of terms :

- rigid designation : The denotation of a function symbol is the same in every world.
- non rigid designation : A function symbol may denote different functions in different worlds.

The existence of terms :

- local terms : For any ground term, its extension belongs to every domain.
- non local terms : The extension of a ground term does not have to belong to every domain.

By choosing different combinations of these variants, several quantified modal logics (QML) are defined : they are called DDE (Domain/ Designation/ Existence) variants of QML.

1.2 Tableaux systems

Tableaux are refutation systems. In order to prove a formula F, one tries to refute $\neg F$, producing a contradiction. This is done by finding a substitution allowing every leaf of the tableau to contain two complementary literals. This procedure involves expanding $\neg F$ (by applying expansion rules) until complementary literals are found (or no expansion rule can be applied any more). In a tableau proof, such a construction takes the form of a tree. If all the branches of the tree can be closed by a same substitution then the tree is said to be closed and the sentence $\neg F$ is refutable.

1.3 The tableau systems Tab_{FV}

The tableau system Tab_{FV} is defined in [CMC00, CMC01]. By lake of space, we will do not recall its expansion rules here.

2 The exploration strategy

Each node of a Tab_{FV} tableau tree is a set of formulae. The calculus Tab_{FV} is intrinsically non-deterministic, i.e. a node might be expanded by means of several rules. Hence, it is crucial to define a fair and complete strategy of rule applications if one wants to implement it.

Different properties are associated to expansion rules. We have classified rules as follows. A rule is:
• **static** if its application does not lead to a loss of information. Otherwise, the rule is **dynamic**. A dynamic rule application correspond to an access to a new world. Typical dynamic rules are those allowing one to expand formulae of the form $\Diamond A$.
• **"looping"** if it can be activated an unlimited number of times by the same formula (same = structurally equal). A typical looping rule is the rule called "γ-rule" allowing to treat universal quantification. A looping rule is static.
• **branching** when it produces more than one expansion, hence generates two branches. Only one rule is branching in the considered system : it is the rule allowing to expand disjunction. This rule is static.

Each rule enjoys one and only one of these four properties:
(1) static and not looping and not branching, (2) looping, (3) branching, or (4) dynamic.

A *block* is a sequence of rule applications. A block is built by stages:
(stage 1) Any static not looping and not branching rule is applied as far as possible until no such a rule is applicable.
(stage 2) For each formula, the corresponding looping rule is applied just once in the block[1].
(stage 3) Any branching rule is applied as far as possible until no such a rule is applicable.
(stage 4) Even if several dynamic rules can be applied, only one dynamic rule is applied and this ends the construction of the block.

[1] but see the following discussion of the backtracking mechanism

Each tableau branch is expanded *block by block*. The end of any block corresponds to a dynamic rule application i.e. to an access to a new world.

The tree exploration is by depth-first search.

Two bounds are defined : B1 is the maximum number of explored worlds (intuitively, the maximum number of dynamic rule applications, generating new worlds in the branch) and B2 is the maximum number of the γ-rule applications to a same formula in any given world (intuitively, the maximum number of instanciations for a universal formula in any given world). These bounds are needed because a branch, in principle, could be expanded infinitely by an unlimited use of γ and dynamic rules applications.

In order to know whether a branch is closed, we have to test if a substitution which closes the branch exists. This test is always done before building a new block (and when the branch can not be expanded anymore).

We have to deal with two types of choice points: (type A) "which dynamic rule apply in stage 4?" and (type B) "which substitution to choose in order to eventually close all the branches of the tree?". A branch cannot be expanded any more when the maximal exploration bounds are reached or when no more rule can be applied to it. A branch closure fails when the branch cannot be expanded any more and no substitution can close it.

When the current branch closure fails, the applications of any given static rule can not be the cause of failure (by the static rule definition). There can be different reasons of failure : more static rules have to be applied in stages 2 and 3 [2], or a dynamic rule is a bad choice in a stage 4 of a block (corresponding to a bad choice of type A), or the substitution chosen to close the previous branches fails to close the current branch, corresponding to a bad choice of type B.

If the branch closure fails, we backtrack always before stage 4 : this allows us not to loose the previous static rule applications and apply a looping rule one more time. This is essential to the completeness of the strategy and allows us to apply a looping rule to the same formula k times (where k is a bound provided by the user) in a given block, as an effect of backtracking.

The first choice of backtracking is the most recent choice point of type A. If the current branch can not be closed by backtracking over all the choice points of type A in the branch, then one tries to find another substitution on the previous branch (corresponding to backtrack on a choice of type B).

[2] That is, a γ-rule needs to be applied again to the same formula.

In [Thi01], we prove that our strategy is sound and complete, provided that, by iterative deepening, the bounds B1 and B2 on the tableau construction are incremented as far as needed (this correspond to use an "unbounded" version of the strategy).

The choice of applying deterministic rules (in our case : static rules) before a non-deterministic (a dynamic) one is classical.

In principle, a γ-rule might need to be applied an unlimited number of times to a same formula in a given world, in order to achieve completeness of the proof search. A classical way to deal with this problem is to set an upper bound (B2 in our case) on the number of γ-rule applications to a same formula in a given world [Fit96].

The specific difficult issue in our case, is to combine these two classical approaches. Applying B2 times a γ-rule to a same formula in a given world leads immediately to an explosion of the search space. In our implementation, the first time the prover tries to refute $\neg F$, it applies just once a γ-rule to a same formula in each world. The fact that the number of γ-rule applications may be increased up to the bounds B2 is an effect of backtracking.

The advantage of such a choice is that if a proof exists where the γ-rules are applied a number of times inferior to the upper bound B2, such a proof is found by the prover.

3 ${Tab_{FV}}^{Prover}$ description

3.1 Short description

The aim of ${Tab_{FV}}^{Prover}$ is to detect whether a given sentence is a theorem of a given QML. Of course, proofs are explored up to a given depth, entered by the user. The strategy explained previously is ${Tab_{FV}}^{Prover}$'s strategy.

3.2 User part

In our prototype, the user indicates the sentence to prove, the propositional base (according to the accessibility relation) of the considered logic, the DDE variant, two bounds respectively on the number of explored worlds and the number of looping rule applications to a same formula.

Otherwise, the proof-search is completely automatic.

Tab_{FV} (thus, ${Tab_{FV}}^{Prover}$ too) deals with the propositional bases K, D, K4, T and S4 (see [Eme90] for a formal definition of the logics according to

their accessibility relation).

3.3 Implementation : language and architecture

We have implemented Tab_{FV} in Objective Caml. It is accessible on line at `http://www.lri.fr/~thion/Proto/proto.php`. One just needs a standard web browser to use it. The Objective Caml program is compiled on the LRI[3] server. The user formulates a query by filling the form in the web page.

A *php* function calls the execution of the program on the server and the answer of the program is given to the user in a frame under the form.

In order to help the user, a pre-defined set of sentences to test and help topics supplement the form.

3.4 Purpose of the tool

The tool ${Tab_{FV}}^{Prover}$ will be part of more important tool allowing to test the constraint preservation in data bases.

Acknowledgments

The author wishes to thank Serenella Cerrito who is the adviser of her *Stage de DEA* (first year of graduate studies), and Marta Cialdea Mayer for many helpful e-mail discussions.

References

[CMC00] Marta Cialdea Mayer and Serenella Cerrito. Variants of first-order modal logics. In R. Dyckhoff, editor, *Proc. of Tableaux 2000*, volume 1847 of *LNAI*, pages 175–189. Springer Verlag, 2000.

[CMC01] Marta Cialdea Mayer and Serenella Cerrito. Ground and free-variable tableaux for variants of quantified modal logic. *Studia Logica, special issue on Analytic Tableaux*, 69, 2001.

[Eme90] E. Allen Emerson. *Handbook of theoretical computer science*, chapter Temporal and modal logic, pages 996–1070. ELsevier Science Publishers B.V., 1990.

[Fit88] M. Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 4:191–213, 1988.

[Fit96] M. Fitting. *First-Order Logic and Automated Theorem Proving (Second Edition)*. Springer, 1996.

[Kri63] S. A. Kripke. Semantical analysis of modal logic *i*, normal propositional calculi. *Zitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[Thi01] V. Thion. Logiques modales et contraintes dynamiques en bases de données. Rapport de stage de DEA, 2001.

[3]Laboratoire de Recherche Informatique, *Université Paris-Sud*, France