MProlog: An Extension of Prolog for Modal Logic Programming

Linh Anh Nguyen

Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We introduce our implemented modal logic programming system MProlog. This system is written in Prolog as a module for Prolog. Codes, libraries, and most features of Prolog can be used in MProlog programs. The system contains a number of built-in SLD-resolution calculi for modal logics, including calculi for useful multimodal logics of belief.

Modal logics can be used to reason about knowledge, belief, actions, etc. A number of authors have proposed modal extensions for logic programming; see [7] for a survey and [6, 1, 3, 4] for later works. There are two approaches for modal logic programming: the direct approach and the translational approach. The first approach directly uses modalities, while the second one translates modal logic programs to classical logic programs.

Despite that the theory of modal logic programming has been studied in a considerable number of works, it has not received much attention in practice. But if we want to use modal logics for practical applications, then modal logic programming deserves for further investigations, especially in practical issues.

In [3, 4], we proposed a general framework for developing semantics of modal logic programs and gave sound and complete SLD-resolution calculi for a number of modal logics, including calculi for useful multimodal logics of belief. Our framework uses a direct approach for handling modalities and does not require any special restriction on occurrences of modal operators (the used language is as expressive as the general modal Horn fragment). Starting from the purely logical formalism of [3, 4], we have built a real system called MProlog [5] for modal logic programming. The implemented system adds extra features to the logical formalism in order to increase usefulness of the language. It is written in Prolog as a module for Prolog and can run in SICStus Prolog and SWI-Prolog. Codes, libraries, and most features of Prolog can be used in MProlog programs in a pure way. This gives MProlog capabilities for real applications. MProlog has been designed to obtain high effectiveness and flexibility. For effectiveness, classical fragments are interpreted by Prolog itself and a number of options can be used for MProlog to restrict the search space. For flexibility, there are three kinds of predicates (classical, modal, and *dum*[1]) and we can use and mix different calculi in an MProlog program.

[1] i.e. classical predicates which are defined using modal formulae

MProlog has a very different theoretical foundation than the implemented Molog system [2]. In MProlog, a labeling technique is used for existential modal operators instead of skolemization. We also provide and use new technicalities like normal forms of modalities or pre-orders between modal operators. MProlog also eliminates drawbacks of Molog (e.g., MProlog gives computed answers).

Our system contains a number of built-in SLD-resolution calculi for modal logics, including calculi for multimodal logics intended for reasoning about multi-degree belief, for use in distributed systems of belief, or for reasoning about epistemic states of agents in multi-agent systems. SLD-resolution calculi for MProlog are specified using our framework given in [3, 4] and written in Prolog. They contain rules (used as meta-clauses) for handling properties of the base modal logic, definitions of auxiliary predicates, and definitions for a number of required predicates (e.g., to specify the normal form of modalities).

In MProlog, modalities are represented as lists, e.g., $\Box_i \Diamond_j q(x)$ is represented as $[bel(I), pos(J)] : q(X)$, and $\Box_x god\_exists \leftarrow christian(x)$ is represented as $[bel(X)] : god\_exists$:- $christian(X)$. Notations of modal operators depend on how the base SLD-resolution calculus is defined. As another example, for MProlog-$\Box$ [4], which disallows existential modal operators in program clauses and goals, we represent $\Box_{i_1} \ldots \Box_{i_k}$ as $[I1, \ldots, Ik]$.

Syntactically, an MProlog program is a Prolog program. Each modal clause in an MProlog program is of one of the following forms:

Context : (Head :- Body). or *Head :- Body.*

where *Context* is a list representing a modality, *Head* is of the form E or $M : E$, E is a classical atom, and M is a list containing one modal operator.

In summary, our MProlog system is a tool for experimenting with applications of modal logic programming to AI. It is also a tool for developing and experimenting with new SLD-resolution calculi for modal logic programming.

References

1. M. Baldoni, L. Giordano, and A. Martelli. A framework for a modal logic programming. In *Joint Int. Conf. and Symp. on Logic Prog.*, p.52–66. MIT Press, 1996.
2. L. Fariñas del Cerro. Molog: A system that extends Prolog with modal logic. *New Generation Computing*, 4:35–50, 1986.
3. L.A. Nguyen. A fixpoint semantics and an SLD-resolution calculus for modal logic programs. *Fundamenta Informaticae*, 55(1):63–100, 2003.
4. L.A. Nguyen. Multimodal logic programming and its applications to modal deductive databases. *manuscript (served as a technical report), available on Internet at http://www.mimuw.edu.pl/~nguyen/papers.html*, 2003.
5. L.A. Nguyen. Source files, calculi, and examples of MProlog. *Available on Internet at http://www.mimuw.edu.pl/~nguyen/mprolog*, 2004.
6. A. Nonnengart. How to use modalities and sorts in Prolog. In C. MacNish, D. Pearce, and L.M. Pereira, editors, *Proceedings of JELIA'94*, LNCS 838, pages 365–378. Springer, 1994.
7. M.A. Orgun and W. Ma. An overview of temporal and modal logic programming. In D.M. Gabbay and H.J. Ohlbach, editors, *Proc. First Int. Conf. on Temporal Logic*, LNAI 827, pages 445–479. Springer-Verlag, 1994.