MSPASS: Modal Reasoning by Translation and First-Order Resolution

Ullrich Hustadt[1] and Renate A. Schmidt[2]

[1] Centre for Agent Research and Development, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom, `U.Hustadt@doc.mmu.ac.uk`
[2] Department of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom, `schmidt@cs.man.ac.uk`

Abstract. MSPASS is an extension of the first-order theorem prover SPASS, which can be used as a modal logic theorem prover, a theorem prover for description logics and a theorem prover for the relational calculus.

1 MSPASS

MSPASS [17] is an enhancement of the first-order theorem prover SPASS with a translator of modal formulae, formulae of description logics, and formulae of the relational calculus. SPASS [19, 20] is one of the fastest and most sophisticated theorem provers for first-order logic with equality, and it's performance compares well with special purpose theorem provers for modal logics, description logics and first-order logic [7, 11, 18].

The input language of SPASS was extended to accept as input also modal, relational and description logic formulae. Modal formulae and description logic formulae are built from a vocabulary of propositional symbols of two disjoint types, namely, propositional (Boolean or concept) and relational (role). The repertoire of logical constructs includes:

– the standard Boolean operators on both propositional and relational formulae: true, false, not, and, or, implies (subsumed by), implied (subsumes), equiv,
– multi-modal modal operators, permitting complex relational parameters: dia and box (synonyms are some and all, the existential and universal role restriction operations of description logics), as well as domain and range,
– the relational operators: comp (composition), sum (relative sum), conv (converse), id (identity), div (diversity), and
– test (test), domrestr (domain restriction) and ranrestr (range restriction).

MSPASS supports modal axiom formulae which are true in every possible worlds (generalised terminological axioms for both concepts and roles). In addition, it is possible to specify additional frame properties, or any other first-order restrictions on the translated formulae.

2 Architecture and Algorithm

Reasoning in MSPASS is performed in three stages: (i) translation of a given set of modal and relational formulae into a set of first-order formulae, (ii) transformation into clausal form, and (iii) saturation-based resolution.

In the current implementation of MSPASS the available translation methods include:

- the standard relational translation method,
- the functional translation method [1, 5, 14],
- the optimised functional translation method [15] including a variation defined in terms of n-ary predicates [7], as well as
- the semi-functional translation method [12].

The implementation of the relational translation method is most general and applies to the language described in Section 1. Some restrictions apply to the other methods. The functional translation, optimised functional and semi-functional translation methods are available only for multi-modal $\mathrm{K}_{(m)}$, and extensions with serial modalities.

Part of MSPASS is a fast converter of first-order formulae into clausal form with special features such as optimised and strong Skolemisation, and an improved implementation of renaming [13].

The inference engine of MSPASS is an implementation of a saturation-based resolution and superposition calculus with simplification [2]. In particular,

- it uses ordered resolution, and ordered superposition with selection,
- it supports splitting and branch condensing (splitting amounts to case analysis while branch condensing resembles branch pruning or backjumping),
- it has an extensive set of reduction and simplification rules, and
- it supports dynamic sort theories by additional inference and reduction rules.

Ordered inference, splitting, and condensing are of particular importance concerning the performance for satisfiable formulae, and for randomly generated formulae unit propagation and branch condensing are important as well.

Using certain flag settings MSPASS is known to provide decision procedures for a range of logics. More specifically:

- For the relational translation of $\mathrm{K}_{(m)}(\cup, \cap, ^-, ^\smile)$ [3] and the description logic $\mathcal{ALB}$ [9]. $\mathrm{K}_{(m)}(\cup, \cap, ^-, ^\smile)$ is the multi-modal logic defined over relations closed under union, intersection, complementation and converse. It includes logics such as the basic tense logic K_t, Humberstone's logic of inaccessibility, Boolean modal logic, as well as fragments of Tarski's relational calculus.
- For the relational translation of $\mathrm{K}_{(m)}(\cup, \cap, ^-, ^\smile)$ and $\mathcal{ALB}$ where the relations satisfy additional first-order restrictions including reflexivity, irreflexivity, seriality, symmetry, density, relational inclusion, etc [3, 9].
- For the optimised functional and semi-functional translation of multi-modal $\mathrm{K}_{(m)}$ in which modalities may be serial [6, 16].
- For the guarded fragment and other solvable fragments [4, 8].

3 Implementation

MSPASS is implemented in ANSI C and differs from SPASS only in the extended input language and the translation routines.

The inference loop in MSPASS is controlled by two sets of clauses: The set of worked-off clauses and the set of usable clauses. At the beginning all input clauses are usable. The theorem prover starts by choosing, according to some heuristic, a clause from the usable clause set and moving it to the worked-off clause set. Then all inference rules are applied to the chosen clause and clauses from the worked-off clause set. The derived clauses are subjected to the reduction rules, for example, subsumption deletion and clause reduction, and non-redundant clauses are added to the usable clause set. If the empty clause is derived or the set of usable clauses is empty, the theorem prover stops, having derived a contradiction or a satisfiable clause set. Otherwise, the next clause is chosen from the usable clause set, moved to the worked-off clause set, and the loop is repeated. The heuristic used by MSPASS chooses a clauses with the minimal number of symbols.

Among the inference rules, splitting has priority. If splitting is applicable to a clause, a heuristic is used to determine which subclauses are generated and in which order they are considered. The implementation of splitting used in this version of MSPASS does not split Horn clauses, and the heuristic used will simply branch on the first (in the order literals are stored internally) positive literal of the chosen clause. These heuristics are very different to those employed by other provers for modal and description logics, and most probably the heuristics of MSPASS are not adequate for the classes of formulae under consideration here.

As most first-order theorem provers, MSPASS uses an indexing data structure to store clauses with the intention of speeding up access to clauses with particular characteristics. To keep the indexing data structure small, structure sharing for terms and atoms is used. That means every occurrence of any subterm of an atom in the clause set exists exactly once and is shared by all superterms containing this subterm. Further implementation details can be found in [20].

4 Special Features

As mentioned in previous sections MSPASS is a reasoner for a large class of modal logics and description logics. Although it does not provide a decision procedure for all the modal logics one may be interested in, for example, PDL or graded modal logic are exceptions, an attractive feature is the possibility to specify arbitrary first-order restrictions. This allows for its use as a flexible tool for the investigation of combinations of interacting non-classical logics or description logics, which have not been been studied in depth before, and in particular, which have not been anticipated by the implementors.

In this context it is useful that, on termination, MSPASS does not only produce a ‘yes’/‘no’ answer, but it also outputs a proof or a saturated set of clauses, if the input problem is unsatisfiable or satisfiable. A finite saturated set of clauses provides a characterisation of a class of models for the input problem.

5 Performance

The tables below present the performance results for MSPASS on the TANCS-2000 comparison problems. The entries in each column have the form t, s, f, where t is the geometric mean of the runtime of s (successfully solved) problems, in 10ms, and f is the number of problems not solved within the time-limit of 7200 CPU seconds. The tests were performed under Sun Solaris 2.6 on a cluster of PCs equipped with 300 MHz Pentium II processors and 256 MB main memory plus 700 MB swap space.

The modal QBF problems in the modal PSPACE division were tested with the optimised functional translation in terms of n-ary predicates (flag settings `-EMLTranslation=2 -EMLFuncNary=1`^{f}), while the converse PDL problems in the global EXPTIME division (the subset without star) and the problems for periodic satisfiability in the global PSPACE division were tested with the relational translation (flag setting `-EMLTranslation=0`^{r}). In all tests ordered resolution with selection of negative literals was used, flag setting `-Ordering=0` with `-Select=1`^{1} for the periodic satisfiability problems and `-Select=2`^{2} for all other problems. The latter results in an ordered hyperresolution like behaviour.

The combination of the relational translation with selection-based resolution or hyperresolution is in general not a decision procedure on the converse PDL and periodic satisfiability problems. In contrast, ordered resolution without selection is a decision procedure for these problems [9]. However, the outcome of the tests performed with ordered resolution was poor. This difference in performance can also be observed for other classes of randomly generated modal formulae [10].

Modal QBF^{f,2}	C10	C20	C30	C40	C50
cnfSSS V4 D4	91 8 -	143 8 -	208 8 -	259 8 -	312 8 -
cnfSSS V4 D6	279 8 -	433 8 -	586 8 -	711 8 -	852 8 -
cnfSSS V8 D4	927 8 -	1516 8 -	1896 8 -	2277 8 -	2627 8 -
cnfSSS V8 D6	3403 8 -	5513 8 -	7017 8 -	8694 8 -	9786 8 -
cnfSSS V16 D4	14820 8 -	22852 8 -	31006 8 -	34324 8 -	41265 8 -
cnfSSS V16 D6	57350 8 -	85530 8 -	107630 8 -	120808 8 -	144932 8 -
cnfLadn V4 D4	3539 8 -	5042 8 -	6034 8 -	6809 8 -	7779 8 -
cnfLadn V4 D6	20708 8 -	37015 8 -	43184 8 -	45131 8 -	47822 8 -
cnfLadn V8 D4	140059 8 -	272815 8 -	354670 8 -	398255 8 -	436614 1 -
cnfLadn V8 D6	612866 3 5	8	8	8	8
cnf V4 D4	5145 4 -	12368 4 -	13575 4 -		
cnf V4 D6	58845 4 -	121394 3 1			
cnf V8 D4	168582 4 -	414258 3 1	4		

Converse PDL (no $*$)^{r,2}	C10	C20	C30	C40	C50
cnfSSS V4 D4	657 8 -	2161 8 -	5292 8 -	10512 8 -	14007 8 -
cnfSSS V4 D6	1325 8 -	4802 8 -	12972 8 -	22660 8 -	30536 8 -
cnfSSS V8 D4	2988 8 -	11802 8 -	33512 8 -	55789 8 -	66524 8 -
cnfSSS V8 D6	6890 8 -	27203 8 -	80986 8 -	105825 8 -	149799 8 -
cnfSSS V16 D4	19096 8 -	61311 8 -	160694 8 -	259507 8 -	334452 8 -
cnfSSS V16 D6	42794 8 -	138737 8 -	344882 8 -	551530 8 -	669153 7 1

PSAT^{r,1}	C20	C30	C40	C50
cnf V4 D1	9 8 -	15 8 -	24 8 -	31 8 -
cnf V4 D2	131 7 1	3082 6 2	6559 2 6	6582 7 1
cnf V8 D1	8 8 -	18 8 -	66 8 -	184 8 -
cnf V8 D2	46 5 2	163144 2 6	8	8
inv V4 D1	114 8 -	373 8 -	275 8 -	171 8 -
inv V4 D2	8	8	8	8
inv V8 D1	51 8 -	5713 8 -	199490 6 2	8
inv V8 D2	8	8	8	8

Reference problems		
QBF-cnf^{f,2}	C20 V2 D2	44 64 -
PSAT-cnf^{r,1}	C32 V4 D1	16 64 -
PSAT-inv^{r,1}	C32 V4 D1	386 64 -

References

1. Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. *J. Logic Computat.*, 2(3):247–297, 1992.
2. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Logic Computat.*, 4(3):217–247, 1994.
3. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic J. IGPL*, 2000. To appear.
4. H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. LICS'99*, pp. 295–303. IEEE Computer Soc., 1999.
5. A. Herzig. *Raisonnement automatique en logique modale et algorithmes d'unification.* PhD thesis, Univ. Paul-Sabatier, Toulouse, 1989.
6. U. Hustadt. *Resolution-Based Decision Procedures for Subclasses of First-Order Logic.* PhD thesis, Univ. d. Saarlandes, Saarbrücken, Germany, 1999.
7. U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *J. Appl. Non-Classical Logics*, 9(4):479–522, 1999.
8. U. Hustadt and R. A. Schmidt. Maslov's class K revisited. In *Proc. CADE-16*, vol. 1632 of *LNAI*, pp. 172–186. Springer, 1999.
9. U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, vol. 1761 of *LNAI*, pp. 192–206. Springer, 2000.
10. U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. To appear in *J. Automated Reasoning*, 2000.
11. U. Hustadt, R. A. Schmidt, and C. Weidenbach. Optimised functional translation and resolution. In *Proc. TABLEAUX'98*, *LNAI* 1397, pp. 36–37. Springer, 1998.
12. A. Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. IJCAI'93*, pp. 80–85. Morgan Kaufmann, 1993.
13. A. Nonnengart, G. Rock, and C. Weidenbach. On generating small clause normal forms. In *Proc. CADE-15*, vol. 1421 of *LNAI*, pp. 397–411. Springer, 1998.
14. H. J. Ohlbach. Semantics based translation methods for modal logics. *J. Logic Computat.*, 1(5):691–746, 1991.
15. H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. *J. Logic Computat.*, 7(5):581–603, 1997.
16. R. A. Schmidt. Decidability by resolution for propositional modal logics. *J. Automated Reasoning*, 22(4):379–396, 1999.
17. R. A. Schmidt. MSPASS, 1999. `http://www.cs.man.ac.uk/~schmidt/mspass/`.
18. G. Sutcliffe and C. B. Suttner. The CADE-14 ATP system competition. *J. Automated Reasoning*, 21(1):99–134, 1998.
19. C. Weidenbach. SPASS, 1999. `http://spass.mpi-sb.mpg.de`.
20. C. Weidenbach. SPASS: Combining superposition, sorts and splitting. In *Handbook of Automated Reasoning.* Elsevier, 2000. To appear.