MSPASS: Subsumption Testing with SPASS

Ullrich Hustadt[1] and **Renate A. Schmidt**[1] and **Christoph Weidenbach**[2]
[1]Department of Computing, Manchester Metropolitan University,
Chester St., Manchester M1 5GD, United Kingdom
{U.Hustadt, R.A.Schmidt}@doc.mmu.ac.uk
[2]Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany
weidenb@mpi-sb.mpg.de

MSPASS is an enhancement of the first-order theorem prover SPASS [17; 18; 19; 20] with a translator of modal formulae, formulae of description logics, and formulae of the relational calculus into first-order logic with equality. SPASS is one of the fastest and most sophisticated existing first-order theorem provers and its performance compares well with special purpose theorem provers for modal logics, description logics and first-order logic, in general [7; 9; 16].

Input language

The input language of SPASS was extended to accept, in addition to first-order formulae, also modal, relational and description logic formulae as input. Description logic and modal formulae are built from a vocabulary of propositional symbols of two disjoint types, namely, type Boolean (concept) and type relational (role). These need to be declared in advance as predicates with zero arity. The repertoire of logical constructs includes:

- the standard Boolean operators on concepts and roles: true, false, not, and, or, impl (subsumed by), implied (subsumes), equiv,
- existential and universal role restriction operations: some and all, as well as domain and range,
- additional role operators: comp (composition), sum (relative sum), conv (converse), id (identity), div (diversity), and
- test (test), domrestr (domain restriction) and ranrestr (range restriction).

MSPASS supports terminological axioms in their most general form, which includes generalised inclusion and equivalence axioms, for both concepts and roles. In addition, because the first-order symbol associated by the translation mapping to a propositional symbol can be declared, it is possible to specify ABox statements as well as any first-order restrictions on concepts and roles.

Translation mappings

A number of different translation methods are available in MSPASS. These include:

- the standard relational translation method (which is determined by the usual semantics of the language),
- the functional translation method [1; 4; 13] and
- the optimised functional translation method [14] including a variation defined in terms of n-ary predicates instead of unary predicates and paths [6; 7], as well as
- the semi-functional translation method [10; 11].

In the current implementation the standard relational translation method is most general and applies to the language described above. Some restrictions apply to the other methods. The functional translation method and semi-functional translation methods are available only for $\mathcal{ALC}$, possibly with serial (total) roles, plus ABox statements, terminological axioms and general inclusion and equivalence axioms. The optimised functional translation methods are implemented only for $\mathcal{ALC}$, possibly with serial roles.

Features of SPASS

SPASS is a sound and complete theorem prover for first-order logic with equality. It is an implementation of a saturation-based resolution and superposition calculus with simplification [2; 3]. 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.

Of particular importance for satisfiable formulae are ordered inference, splitting, and condensing while for randomly generated formulae unit propagation, and branch condensing are important as well.

Part of SPASS 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 [12].

On termination, SPASS does not only produce a 'yes'/'no' answer, but it also outputs a proof or a saturated set of clauses depending on whether the input problem is unsatisfiable or not. A finite saturated set of clauses provides a characterisation of a class of models for the input problem.

For certain logics and using certain flag settings MSPASS is known to provide decision procedures. More specifically:

- For the relational translation of description logics below $\mathcal{ALB}$ and including $\mathcal{ALB}$ [8]. $\mathcal{ALB}$ is the extension of $\mathcal{ALC}$ with role formulae expressed in terms of the top and bottom role, conjunction, disjunction, negation and converse.
- For the optimised functional translation of $\mathcal{ALC}$ with or without serial, reflexive or transitive roles [15].
- For the semi-functional translation of $\mathcal{ALC}$ with or without serial, reflexive, symmetric or transitive roles [5].

Implementation

Both SPASS and MSPASS are implemented in ANSI C. The translation routines of MSPASS have alpha status (there are plans to change the input syntax slightly).

Performance

Extensive testing has been done with different versions of SPASS but as yet not with MSPASS. The tests which have been performed and published thusfar have been for $\mathcal{ALC}$ or multi-modal $K_{(m)}$ with or without reflexive and transitive modalities [6; 7; 9]. In all cases the translation employed was the optimised functional translation in terms of n-ary predicates.

Availability

SPASS as well as MSPASS are available from

`http://spass.mpi-sb.mpg.de/`

where also interactive use of the systems is possible.

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] L. Bachmair and H. Ganzinger. Equational reasoning in saturation-based theorem proving. In *Automated Deduction—A Basis for Applications. Volume I*, chapter 11, pages 353–397. Kluwer, 1998.

[4] A. Herzig. *Raisonnement automatique en logique modale et algorithmes d'unification.* PhD thesis, Univ. Paul-Sabatier, Toulouse, 1989.

[5] U. Hustadt. *Resolution-Based Decision Procedures for Subclasses of First-Order Logic.* PhD thesis, Univ. d. Saarlandes, Saarbrücken, 1999. Submitted.

[6] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logics. In *Proc. IJCAI'97*, pages 202–207. Morgan Kaufmann, 1997.

[7] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. To appear in *J. Appl. Non-Classical Logics*, 1998.

[8] U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *First-Order Theorem Proving—FTP'98*, *LNAI*. Springer, 1999. To appear.

[9] U. Hustadt, R. A. Schmidt, and C. Weidenbach. Optimised functional translation and resolution. In *Proc. TABLEAUX'98*, volume 1397 of *LNAI*, pages 36–37. Springer, 1998.

[10] A. Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. IJCAI'93*, pages 80–85. Morgan Kaufmann, 1993.

[11] A. Nonnengart. *A Resolution-Based Calculus for Temporal Logics.* PhD thesis, Univ. d. Saarlandes, Saarbrücken, 1995.

[12] A. Nonnengart, G. Rock, and C. Weidenbach. On generating small clause normal forms. In *Automated Deduction: CADE-15*, volume 1421 of *LNAI*, pages 397–411. Springer, 1998.

[13] H. J. Ohlbach. Semantics based translation methods for modal logics. *J. Logic Computat.*, 1(5):691–746, 1991.

[14] 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.

[15] R. A. Schmidt. Decidability by resolution for propositional modal logics. *J. Automated Reasoning*, 22(4):379–396, 1999.

[16] G. Sutcliffe and C. B. Suttner. The CADE-14 ATP system competition. *J. Automated Reasoning*, 21(1):99–134, 1998.

[17] C. Weidenbach. Spass: Version 0.49. *J. Automated Reasoning*, 18:247–252, 1997.

[18] C. Weidenbach and et. al. System description: SPASS version 1.0.0. In *Automated Deduction: CADE-16*, *LNAI*. Springer, 1999. To appear.

[19] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER, version 0.42. In *Automated Deduction: CADE-13*, volume 1104 of *LNAI*, pages 141–145. Springer, 1996.

[20] C. Weidenbach, C. Meyer, C. Cohrs, T. Engel, and E. Keen. SPASS v0.77. *J. Automated Reasoning*, 21(1):113–113, 1998.