System Description: E-KRHyper

Björn Pelzer and Christoph Wernhard

Universität Koblenz-Landau, Koblenz, Germany
{bpelzer, wernhard}@uni-koblenz.de

Abstract. The E-KRHyper system is a model generator and theorem prover for first-order logic with equality. It implements the new E-hyper tableau calculus, which integrates a superposition-based handling of equality into the hyper tableau calculus. E-KRHyper extends our previous KRHyper system, which has been used in a number of applications in the field of knowledge representation. In contrast to most first order theorem provers, it supports features important for such applications, for example queries with predicate extensions as answers, handling of large sets of uniformly structured input facts, arithmetic evaluation and stratified negation as failure. It is our goal to extend the range of application possibilities of KRHyper by adding equality reasoning.

1 Introduction

E-*KRHyper* is a theorem proving and model generation system for first-order logic with equality. It is an implementation of the E-*hyper tableau calculus* [1], which integrates a superposition-based handling of equality [2] into the hyper tableau calculus [3]. If E-KRHyper terminates without finding a refutation, it leaves a finite set of positive unit clauses representing a model of the input. Continued operation effects that alternative models are enumerated, allowing the use of E-KRHyper as a model generator for answer set computation.

E-KRHyper is an extended version of our KRHyper system [4], which is based on the original hyper tableau calculus and therefore lacks a dedicated mechanism for equality reasoning. So far KRHyper has been used as an embedded knowledge processing engine in several applications including content composition for e-learning [5, 6], document management [7], database schema processing [8], semantic information retrieval [9], ontology reasoning [10], and planning [11]. An excerpt of KRHyper has been ported to Mobile Java and is employed for user profile matching on mobile devices [12]. We intend to further this usage with the enhanced reasoning capabilities of our upgraded system. This includes reasoning in modal and description logics, which is only possible in a restricted way with the original KRHyper [13], and which will allow a more accurate modelling of application domains.

2 Language, Usage and Availability

E-KRHyper accepts formulas of first order logic in clausal form. The system supports several language extensions, including stratified negation as failure and a stratified set abstraction construct. The arithmetic constant types, evaluable arithmetic functors and arithmetic built-ins specified in the ISO standard for Prolog are provided. The input syntax is the *Protein* format, which is supported by the *TPTP* tools. The syntax of in- and output complies with ISO standard Prolog. Proofs of refutations and derivations of facts in models can be output as terms which can be visualized with the *Graphviz* tool. The system is implemented in the functional/imperative language *OCaml* with additional preprocessing scripts in *SWI-Prolog*. E-KRHyper runs on Unix and MS-Windows platforms and is available under the GNU Public License from the E-KRHyper website at `http://www.uni-koblenz.de/~bpelzer/ekrhyper`.

3 E-Hyper Tableaux

An E-hyper tableau [1] is a tree whose nodes are labeled with clauses and which is built up by the application of the inference rules of the E-hyper tableau calculus. The calculus rules are designed such that most of the reasoning is performed using positive unit clauses. A branch can be extended with new clauses that have been derived from the clauses of that branch.

A positive disjunction can be used to split a branch, creating a new branch for each disjunct. No variables may be shared between branches, and if a case-split creates branches with shared variables, then these are immediately substituted by ground terms. The grounding substitution is arbitrary as long as the terms in its range are *irreducible*: the branch being split may not contain a positive equational unit which can simplify a substituting term, i.e. rewrite it with one that is smaller according to a reduction ordering [2, 15]. When multiple irreducible substitutions are possible, each of them must be applied in consecutive splittings in order to preserve completeness.

Redundancy rules allow the detection and removal of clauses that are redundant with respect to a branch.

4 Model Generation and Theorem Proving Method

The E-hyper tableau is generated depth-first, with E-KRHyper always working on a single branch. Refutational completeness and a fair search control are ensured by an iterative deepening strategy with a limit on the maximum term weight of generated clauses.

E-KRHyper maintains the clauses on the working branch grouped into two sets, the first containing positive non-equational units and the second containing equational units and clauses that include negative literals. A third set is used to maintain positive non-unit clauses. If all other inference possibilities have been exhausted, the spliting rule picks a clause from this set and extends the working branch by attaching a child for each disjunct to its leaf node. One of the resulting branches is then selected as the new working branch.

If the computation of a branch reaches a fixed point, then a model has been found. If on the other hand a contradiction within a branch is detected, then that branch is abandoned, and the computation backtracks to the next branch. If there is no next branch, computation halts with the result that there is no [more] model.

$$
\boxed{\begin{aligned}
C_1 &= a \simeq b. \\
C_2 &= q(a). \\
C_3 &= r(d). \\
C_4 &= p(x) \vee c \simeq d \leftarrow q(b).
\end{aligned}}
$$

Fig. 1. Example Input

A model is represented by a set of positive unit clauses. These correspond to a convergent rewrite system that is complete with respect to the equational theory represented by the set of input clauses [1]. For example, consider the input clauses displayed in Fig. 1. At the first fixed point of E-KRHyper's derivation, the branch contains the positive unit clauses $\{C_1, C_2, C_3, p(x)\}$, which corresponds to the model $M_1 = \{a \simeq b, q(a), r(d), p(x)\}$. The second and final fixed point corresponds to the model $M_2 = \{q(a), r(c), c \simeq d\}$ – note that C_3 has been simplified into the atom $r(c)$.

Optionally two refinements of theorem proving methods based on model generation are employed: *level cut* [3], a form of dependency directed backtracking, and *complement splitting* [17], which also can be used for the computation of minimal models. The *hyper extension* inference from the original hyper tableau calculus is equivalent to a series of E-hyper tableau calculus inference applications. Therefore the implementation of the hyper extension in KRHyper by a variant of semi-naive evaluation [16] is retained in E-KRHyper, where it serves as a shortcut inference for the resolution of non-equational literals.

5 Comparison to KRHyper

Both KRHyper and E-KRHyper are written in OCaml. The integration of the new calculus in E-KRHyper has required approximately 10,500 lines of additional code compared to KRHyper, representing a size increase of 79 percent. Apart from implementing the new inference rules, it was also necessary to modify a number of original operations. KRHyper only ever adds positive unit clauses to its hyper tableaux, and the indexing is similarly confined to positive units. Also, there is no support for destructive tableau modifications, as the original calculus does not include any such operations. In E-KRHyper the clause indexing has been extended to cover the full range of clauses, and both the derivation loop and the indexing take into account the dynamically growing and shrinking clause sets of the new calculus. On problems without equality, the changes result in E-KRHyper being 24 percent slower than KRHyper.

6 Related Systems and Performance Evaluation

The SPASS system is a superposition-based theorem prover for first-order logic, that however cannot straightforwardy be used for model computation. Like E-KRHyper, SPASS splits on disjunctions. SPASS can only split when the resulting parts are variable disjoint, though. This inability to split on all disjunctions is responsible for failing to decide certain classes of formulas that are decided by E-KRHyper [1].

The basic concept of theorem proving by model generation, as employed in E-KRHyper, stems from *Satchmo* [17]. *Satchmo Compiler* [19] and *MGTP* [20] have been earlier efforts to implement such a system efficiently. GEO [21] is a recent system for theorem proving and computation of finite models in first order logic, which, like E-KRHyper, works by integrating equality processing into model based search. The approach in GEO is not based on superposition and has no redundancy treatment, and while the splitting method is similar to the one in E-KRHyper, the number of eligible grounding substitutions is not limited to those that are irreducible. *Smodels* [22] and *DLV* [23] are systems which efficiently compute stable models but can handle first-order features such as nested terms and nonground terms only in very restricted ways.

We have tested E-KRHyper on several problem groups eligible for the CASC 2006 [24]. The tests were carried out on a 1.5 GHz Pentium M computer with 1.5 GB RAM and a timeout limit of 400 seconds. Table 1 shows the results for those problems finally selected for the competition. As a comparison the official competition results of the Otter 3.3 system [25] are listed as well.[1] In comparision with the competition entrants, E-KRHyper ranks in the middle for Horn and Non-Horn problems with equality (HEQ and NEQ) and in the lower ranges for Non-Horn problems without equality (NNE). For unit equation problems (UEQ) the system is uncompetitive. E-KRHyper retains the general characteristics of KRHyper and thus performs well on certain problem classes without equality [4]. So far the development of the KRHyper line has focused more on the application possibilities than on competition performance, but we hope to optimize the operation in the future. More detailed information and test results are provided on the E-KRHyper website.

Problem class	NNE	HEQ	NEQ	UEQ
Number of problems	20	20	70	100
Solved by E-KRHyper	6	9	18	2
Solved by Otter 3.3	10	12	20	28

Table 1. Results on CASC J3 problems

7 Conclusion

KRHyper has successfully been deployed in real-world applications for knowledge representation. However, its lack of dedicated equality handling has been a limitation in certain areas like reasoning with description logics. The implementation of the new calculus with equality in E-KRHyper has cleared this obstacle

[1] The Otter system represents the state of the art in first-order theorem proving around 1996 and regularly participates in the CASC to provide a stable benchmark.

and opened the way for new integration opportunities. Given that the field of automated theorem proving has come to be dominated by saturation based systems in recent years, we also hope that E-KRHyper will be a first step towards an efficient and competitive tableau based theorem prover with equality.

Acknowledgements. The authors wish to thank Peter Baumgartner and Ulrich Furbach for opening to them the world of model based deduction with hyper tableaux and for their help in implementing the new calculus, as well as the anonymous reviewers for their constructive suggestions on improving this paper.

References

1. P. Baumgartner, U. Furbach, B. Pelzer: *Hyper Tableau with Equality.* In: Fachberichte Informatik 12–2007, Universität Koblenz-Landau, 2007.
2. L. Bachmair and H. Ganzinger: Chapter 11: *Equational Reasoning in Saturation-Based Theorem Proving.* In: W. Bibel and P.H. Schmitt, eds., Automated Deduction – A Basis for Applications, vol. I, pp. 352-397, Kluwer, 1998.
3. P. Baumgartner, U. Furbach, I. Niemelä: *Hyper Tableaux.* In: J.J. Alferes, L.M. Pereira, and E. Orlowska, eds., Proc. of the European Workshop on Logics in Artifical Intelligence (JELIA '96), LNAI, Springer, 1996.
4. C. Wernhard: *System Description: KRHyper.* In: Fachberichte Informatik 14–2003, Universität Koblenz Landau, 2003.
5. P. Baumgartner and U. Furbach: *Living Books, Automated Deduction and other Strange Things.* In D. Hutter and W. Stephan, eds., Mechanizing Mathematical Reasoning: Techniques, Tools and Applications - Essays in honour of Jörg H. Siekmann, Springer LNCS volume 2605, pp. 255-274, 2004.
6. P. Baumgartner, U. Furbach, M. Gross-Hardt, and A. Sinner: *Living Book: deduction, slicing, and interaction.* In: J. of Autom. Reasoning, 32(3), pp. 259-286, 2004.
7. P. Baumgartner, U. Furbach, M. Gross-Hardt, T. Kleemann and C. Wernhard: *KRHyper Inside - Model Based Deduction in Applications.* In: Proc. CADE-19 Workshop on Novel Applications of Deduction Systems, 2003.
8. P. Baumgartner, U. Furbach, Ma. Gross-Hardt and T. Kleemann: *Model Based Deduction for Database Schema Reasoning.* In: S. Biundo, T. Frühwirth, and G. Palm, eds., KI 2004, Springer LNCS volume 3238, pp. 168-182, 2004.
9. P. Baumgartner and A. Burchardt: *Logic Programming Infrastructure for Inferences on FrameNet.* In: J.J. Alferes and J. Leite, eds., Proc. JELIA'04, Springer LNCS volume 3229, pp. 591-603, 2004.
10. P. Baumgartner and F.M. Suchanek: *Automated Reasoning Support for First-Order Ontologies.* In: J.J. Alferes, J. Bailey, W. May, and U. Schwertel, eds., Principles and Practice of Semantic Web Reasoning 4th International Workshop (PPSWR 2006), Springer LNAI volume 4187, pp. 18-32, 2006.
11. P. Baumgartner and A. Mediratta: *Improving Stable Models Based Planning by Bidirectional Search.* In: International Conference on Knowledge Based Computer Systems (KBCS), Hyderabad, India, 2004.
12. T. Kleemann, A. Sinner: *KRHyper - In Your Pocket, System Description.* In: R. Nieuwenhuis, editor: Proc. of the 20th International Conference on Automated Deduction, CADE-20 Springer LNCS vol. 3632, 2005.

13. P. Baumgartner and R. Schmidt: *Blocking and Other Enhancements of Bottom-Up Model Generation Methods.* In: U. Furbach and N. Shankar, eds., Third Int. Joint Conference on Automated Reasoning (IJCAR), Springer LNAI volume 4130, 2006.
14. P. Deransart et al.: *Prolog: The standard: reference manual.* Berlin, 1996.
15. R. Nieuwenhuis and A. Rubio: *Paramodulation-based theorem proving.* In: J.A. Robinson and A. Voronkov, eds., Handbook of Automated Reasoning, pp. 371–443, Elsevier and MIT Press, 2001.
16. J.D. Ullman: *Principles of Database and Knowledge-Base Bystems*, Rockville, Maryland, 1989.
17. R. Manthey, F. Bry: *SATCHMO: A theorem prover implemented in Prolog.* E. Lusk et al., eds.: Proc. of the 9th Conf. on Autom. Deduction, pp.415–434, LNCS, 1988.
18. C. Weidenbach: *Combining Superposition, Sorts and Splitting.* In: A. Robinson and A. Voronkov, eds., Handbook of Automated Reasoning, North Holland, 2001.
19. Schütz, H., Geisler, T.: *Efficient model generation through compilation.* In: M. A. McRobbie and J. K. Slaney, eds., Proc. of the 13th Int. Conf. on Automated Deduction, pp. 433–447, LNAI, 1996
20. Hasegawa, R., Fujita H., Koshimura, M.: *MGTP: A Model Generation Theorem Prover — its advanced features and applications.* In: Autom. Reasoning with Analytic Tableaux (Tableaux '97), pp. 1–15, LNAI, 1997
21. H. de Nivelle and J. Meng: *Geometric Resolution: A Proof Procedure Based on Finite Model Search.* In: U. Furbach and N. Shankar, eds., IJCAR 2006: Third International Joint Conference on Automated Reasoning, Springer LNAI, Vol 4130, pp. 303–317, 2006.
22. I. Niemelä and P. Simons: *Smodels – An implementation of the stable model and well-founded semantics for normal logic programs.* In: J. Dix, U. Furbach and A. Nerode, eds., Proc. of the 4th Int. Conf. on Logic Programming and Non-Monotonic Reasoning, pp. 420-429, 1997.
23. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri and F. Scarcello: *The DLV System for Knowledge Representation and Reasoning.* INFSYS RR-1843-02-14, Technische Universität Wien, 2002.
24. G. Sutcliffe and C. Suttner: *The State of CASC.* In: AI Communications, 19(1), pp. 35–48, 2006.
25. W. McCune: *OTTER 3.3 Reference Manual.* Argonne National Laboratory, Argonne, Illinois, ANL/MCS-TM-263, 2003.