Detecting Unsolvable Queries for Definite Logic Programs

Maurice Bruynooghe¹, Henk Vandecasteele¹,
D. Andre de Waal², and Marc Denecker¹

June 23, 1998

Extended abstract of a paper at PLILP'98 [4]

In logic programming, almost no work has been done so far on proving that certain queries cannot succeed. Work in this direction could be useful for queries which seem to be non-terminating. Such queries are not exceptional, e.g. in planning problems. A related field of research is termination analyses, (see [6] for a survey). However, termination conditions are not decidable and automated methods are based on analysing the size of syntactical structures. So there is a substantial class of programs for which automated termination analysis fails. For some of these, loop checking methods [1, 2, 10] monitoring the execution, bring some relief, by pruning some infinitely failing branches. Methods have to choose between pruning too much, causing incompleteness in the search for solutions, and preserving completeness but still allowing some infinite computations. Also an execution mechanism augmented with tabling such as XSB [9] or an approach such as [3] reduces the number of non-terminating queries. So far, there are almost no works on program analysis which attempt to recognise (infinitely) failing queries. Conceptually, there is a simple way to show that a query must fail: find a model of the program in which the query is false. The paper develops two methods for automating the search for a pre-interpretation underlying such a model. A first approach combines abduction and tabulation to direct the search for a pre-interpretation. A second approach considers the abducibles as constraints and uses techniques from finite domain constraint solving [13] to further prune the search. Also the suitability of alternative methods for solving this problem is analysed. The use of a general purpose model generation tool [11, 12] is evaluated. We have also explored whether tools for type inference [7, 5] can show that such queries have an empty success set and whether conjunctive partial deduction [8] can specialise such queries into a trivially failing program.

A number of experiments were done with the two methods we developed and compared with the results of the alternative methods. In general, our approach is doing much better than the general purpose model generation tool FINDER and type inference based on Regular Approximations. The comparison with conjunctive partial deduction is less straightforward. Both approaches seem to be good at a different classes of problems. In fact it could be interesting to apply program specialisation as a pre-processing step: the specialisation may reduce the number of predicates and the number of functors, making the problem easier to solve by our tool (and in extreme cases making it a trivial problem by returning the empty program).

¹Katholieke Universiteit Leuven, B-3001 Heverlee, Belgium. {maurice,henkv,marcd}@cs.kuleuven.ac.be

²Potchefstroom University for Christian Higher Education, S. Africa. BWIDADW@puknet.puk.ac.za

References

- [1] K.R. Apt, R.N. Bol, and J.W. Klop. On the safe termination of Prolog programs. In *Proc. ICLP'89*, pages 353–368, Lisbon, june 1989. MIT Press.
- [2] R.N. Bol, K.R. Apt, and J.W. Klop. An analysis of loop checking mechanisms for logic programs. *Theoretical Computer Science*, 86(1):35–79, august 1991.
- [3] M.P. Bonacina and J. Hsiang. On rewrite programs: semantics and relationship with Prolog. *Journal of Logic Programming*, 14(1&2):155–180, october 1992.
- [4] Maurice Bruynooghe, Henk Vandecasteele, Andre de Waal, and Marc Denecker. Detecting unsolvable queries for definite logic programs. In *Proceeding of PLILP, accepted*, September 1998.
- [5] W. Charatonik and A. Podelski. Directional type inference for logic programs. In *Fifth International Static Analysis Symposium*, LNCS, Pisa, Italy, 1998. to appear.
- [6] D. De Schreye and S. Decorte. Termination of logic programs: the never-ending story. *Journal of Logic Programming*, 19 & 20:199–260, may/july 1994.
- [7] J. P. Gallagher and D.A. de Waal. Fast and precise regular approximations of logic programs,. In P. Van Hentenryck, editor, *Proc. ICLP94*, pages 599–613. MIT Press, 1994.
- [8] Michael Leuschel, Danny De Schreye, and André de Waal. A conceptual embedding of folding into partial deduction: Towards a maximal integration. In Michael Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming JICSLP'96*, pages 319–332, Bonn, Germany, September 1996. MIT Press. Extended version as Technical Report CW 225, K.U. Leuven. Accessible via <http://www.cs.kuleuven.ac.be/~lpai>.
- [9] K. Sagonas, T. Swift, and D. S. Warren. XSB as an efficient deductive database engine. In *Proceedings of SIGMOD 1994 Conf. ACM*. Acm Press, 1994.
- [10] Y. D. Shen. An extended variant of atoms loop check for positive logic programs. *New Generation Computing*, 15(2):187–203, 1997.
- [11] J. Slaney. Finder: Finite domain enumerator system description. Technical Report TR-ARP-2-94, Centre for Information Science Research, Australian National University, Australia, 1994. Also in Proc. CADE-12.
- [12] J. Slaney. Finder - finite domain enumerator - version 3.0 - notes and guide. Technical report, Centre for Information Science Research, Australian National University, Australia, 1997.
- [13] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. The MIT press, 1989.