Hasta-La-Vista: Termination Analyser for Logic Programs

Alexander Serebrenik, Danny De Schreye

Department of Computer Science, K.U. Leuven
Celestijnenlaan 200A, B-3001, Heverlee, Belgium
E-mail: {Alexander.Serebrenik,Danny.DeSchreye}@cs.kuleuven.ac.be

Verifying termination is often considered as one of the most important aspects of program verification. Logic languages, allowing us to program declaratively, increase the danger of non-termination. Therefore, termination analysis received considerable attention in logic programming (see e.g. [7, 8, 10, 16]). Unfortunately, the majority of existing termination analysers, such as TermiLog [15], TerminWeb [7], and cTI [16] are restricted to pure logic programs and thus, leave many interesting real-world examples out of consideration. Therefore, in order to abridge the gap between programming practice and existing termination analysers real-world programming techniques should be considered.

In this paper we present Hasta-La-Vista—a powerful tool for analysing termination of logic programs with integer computations. Hasta-La-Vista extends the constraints-based approach of Decorte *et al.* [9] by integrating the inference algorithm of [19]. Moreover, as explained in [19], in the integer case our approach is not limited to proving termination, but can also infer termination, i.e., find the set of queries terminating for a given program.

System architecture. Conceptually, Hasta-La-Vista consists of three main parts: transformation, constraints generation and constraints solving. As a preliminary step, given a program and a set of atomic queries, type analysis of Janssens and Bruynooghe [14] computes the call set. We opted for a very simple type inference technique that provides us only with information whether some argument is integer or not.

Based on the results of the type analysis the system decides whether termination of the given program can be dependent on the integer computation. In this case, the *adorning* transformation is applied [19]. The aim of the transformation is to discover bounded integer arguments and to make the bounds explicit. Intuitively, if a variable x is known to be bounded from above by n, then $n - x$ is always positive and thus, can be used as a basis for a definition of a *level-mapping* (a function from the set of atoms to the naturals). In order to prove termination we have to show that the level-mapping decreases while traversing a clause. This requirement can be translated into a set of constraints. Finally, this set of constraints is solved and depending on the solution termination is

reported for all queries or for some queries or possibility of non-termination is suspected.

Experimental evaluation. Hasta-La-Vista has been applied to more than 90 examples. In 95% of the cases the analysis was either powerful enough to prove termination for terminating computations, or justly suspected possibility of non-termination. Results of the experimental evaluation are summarised in Table 1. Times were measured on Intel®Pentium®4 with 1.60GHz CPU and 260M memory, running 2.4.20-pre11 Linux. The core part of the implementation was done in SICStus Prolog [20], using its CLP(FD) [5] and CLP(Q) [12] libraries. Type inference of Janssens and Bruynooghe [14] was implemented in MasterProLog [13].

Table 1. Experimental evaluation

Reference	**Number of examples**	**Success rate**	**Maximal time**
Symbolical computations			
Apt and Pedreschi [2]	12	100%	0.05
De Schreye and Decorte [8]	7	85%	0.01
Plümer [18]	31	90%	0.09
Integer computations			
Apt [1], chapter 9	13	100%	0.08
Sterling and Shapiro [21], chapter 8	14	100%	0.02
Various	19	100%	0.27
Total	92	95%	0.24

Hasta-La-Vista turned out to be robust enough to prove termination of such examples as Euler's totient function [11], depth-limited depth-first search [3] and finding all prime numbers up to a given limit [6]. Programs denoted as "various" were collected both from different Prolog textbooks [3, 6, 17] and from Prolog programs collections [4, 11].

Conclusion. We have presented Hasta-La-Vista, a termination analyser for logic programs. To the best of our knowledge, this analyser is unique in being able to prove termination of programs depending on integer computations.

Acknowledgement. We are very grateful to Gerda Janssens for making her type analysis system available for us.

References

1. K. R. Apt. *From Logic Programming to Prolog.* Prentice-Hall International Series in Computer Science. Prentice Hall, 1997.

2. K. R. Apt and D. Pedreschi. Modular termination proofs for logic and pure Prolog programs. In G. Levi, editor, *Advances in Logic Programming Theory*, pages 183–229. Oxford University Press, 1994.
3. I. Bratko. *Prolog programming for Artificial Intelligence*. Addison-Wesley, 1986.
4. F. Bueno, M. J. García de la Banda, and M. V. Hermenegildo. Effectiveness of global analysis in strict independence-based automatic parallelization. In M. Bruynooghe, editor, *Logic Programming, Proceedings of the 1994 International Symposium*, pages 320–336. MIT Press, 1994.
5. M. Carlsson, G. Ottoson, and B. Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. H. Hartel, and H. Kuchen, editors, *Programming Languages: Implementations, Logics, and Programs, 9th International Symposium, PLILP'97, Including a Special Track on Declarative Programming Languages in Education, Southampton, UK, September 3-5, 1997, Proceedings*, volume 1292 of *Lecture Notes in Computer Science*, pages 191–206. Springer Verlag, 1997.
6. W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer Verlag, 1981.
7. M. Codish and C. Taboch. A semantic basis for termination analysis of logic programs. *Journal of Logic Programming*, 41(1):103–123, 1999.
8. 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.
9. S. Decorte, D. De Schreye, and H. Vandecasteele. Constraint-based termination analysis of logic programs. *ACM TOPLAS*, 21(6):1137–1195, November 1999.
10. N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic programs. *Applicable Algebra in Engineering, Communication and Computing*, 12(1-2):117–156, 2001.
11. W. Hett. P-99: Ninety-nine Prolog problems. Available at `http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/`, July 2001.
12. C. Holzbaur. OFAI CLP(Q,R) Manual. Technical Report TR-95-09, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.
13. IT Masters. MasterProLog Programming Environment. Available at `http://www.itmasters.com/`, 2000.
14. G. Janssens and M. Bruynooghe. Deriving descriptions of possible values of program variables by means of abstract interpretation. *Journal of Logic Programming*, 13(2&3):205–258, July 1992.
15. N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. *TermiLog*: A system for checking termination of queries to logic programs. In O. Grumberg, editor, *Computer Aided Verification, 9th International Conference*, volume 1254 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, June 1997.
16. F. Mesnard and U. Neumerkel. Applying static analysis techniques for inferring termination conditions of logic programs. In P. Cousot, editor, *Static Analysis, 8th International Symposium, SAS 2001*, volume 2126 of *Lecture Notes in Computer Science*, pages 93–110. Springer Verlag, 2001.
17. R. A. O'Keefe. *The Craft of Prolog*. MIT Press, 1990.
18. L. Plümer. *Termination Proofs for Logic Programs*, volume 446 of *Lecture Notes in Computer Science*. Springer Verlag, 1990.
19. A. Serebrenik and D. De Schreye. Inference of termination conditions for numerical loops in Prolog. In R. Nieuwenhuis and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conferencerence, Proceedings*, volume 2250 of *Lecture Notes in Computer Science*, pages 654–668. Springer Verlag, 2001.
20. SICS. *SICStus User Manual. Version 3.10.0*. Swedish Institute of Computer Science, 2002.
21. L. Sterling and E. Shapiro. *The Art of Prolog*. The MIT Press, 1994.