Programming with Global Analysis

Manuel Hermenegildo
The CLIP Group
School of Computer Science
Technical University of Madrid
herme@fi.upm.es

Abstract

Global data-flow analysis of (constraint) logic programs, which is generally based on abstract interpretation [7], is reaching a comparatively high level of maturity. A natural question is whether it is time for its routine incorporation in standard compilers, something which, beyond a few experimental systems, has not happened to date. Such incorporation arguably makes good sense only if:

- the range of applications of global analysis is large enough to justify the additional complication in the compiler, and
- global analysis technology can deal with all the features of "practical" languages (e.g., the ISO-Prolog built-ins) and "scales up" for large programs.

We present a tutorial overview of a number of concepts and techniques directly related to the issues above, with special emphasis on the first one. In particular, we concentrate on novel uses of global analysis during program development and debugging, rather than on the more traditional application area of program optimization.

The idea of using abstract interpretation for validation and diagnosis has been studied in the context of imperative programming [2] and also of logic programming. The latter work includes issues such as using approximations to reduce the burden posed on programmers by declarative debuggers [6, 3] and automatically generating and checking assertions [4, 5] (which includes the more traditional type checking of strongly typed languages, such as Gödel or Mercury [1, 8, 9])

We also review some solutions for scalability including modular analysis, incremental analysis, and widening. Finally, we discuss solutions for dealing with meta-predicates, side-effects, delay declarations, constraints, dynamic predicates, and other such features which may appear in practical languages.

In the discussion we will draw both from the literature and from our experience and that of others in the development and use of the CIAO system analyzer. In order to emphasize the practical aspects of the solutions discussed, the presentation of several concepts will be illustrated by examples

run on the CIAO system, which makes extensive use of global analysis and assertions.

References

[1] K. R. Apt and E. Marchiori. Reasoning about Prolog programs: from modes through types to assertions. *Formal Aspects of Computing*, 6(6):743–765, 1994.

[2] F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Programming Languages Design and Implementation'93*, pages 46–55, 1993.

[3] J. Boye, W. Drabent, and J. Małuszyński. Declarative diagnosis of constraint programs: an assertion-based approach. In *Proc. of the 3rd. Int'l Workshop on Automated Debugging–AADEBUG'97*, pages 123–141, Linkoping, Sweden, May 1997. U. of Linkoping Press.

[4] F. Bueno, D. Cabeza, M. Hermenegildo, and G. Puebla. Global Analysis of Standard Prolog Programs. In *European Symposium on Programming*, number 1058 in LNCS, pages 108–124, Sweden, April 1996. Springer-Verlag.

[5] F. Bueno, P. Deransart, W. Drabent, G. Ferrand, M. Hermenegildo, J. Maluszynski, and G. Puebla. On the Role of Semantic Approximations in Validation and Diagnosis of Constraint Logic Programs. In *Proc. of the 3rd. Int'l Workshop on Automated Debugging–AADEBUG'97*, pages 155–170, Linkoping, Sweden, May 1997. U. of Linkoping Press.

[6] M. Comini, G. Levi, M. C. Meo, and G. Vitiello. Proving properties of logic programs by abstract diagnosis. In M. Dams, editor, *Analysis and Verification of Multiple-Agent Languages, 5th LOMAPS Workshop*, number 1192 in Lecture Notes in Computer Science, pages 22–50. Springer-Verlag, 1996.

[7] P. Cousot and R. Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Fourth ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977.

[8] P. Hill and J. Lloyd. *The Goedel Programming Language*. MIT Press, Cambridge MA, 1994.

[9] Z. Somogyi, F. Henderson, and T. Conway. The execution algorithm of Mercury: an efficient purely declarative logic programming language. *JLP*, 29(1–3), October 1996.