A Generic Framework for the Analysis and Specialization of Logic Programs

Germán Puebla[1], Elvira Albert[2], and Manuel Hermenegildo[1,3]

[1] School of Computer Science, Technical U. of Madrid, {german,herme}@fi.upm.es
[2] School of Computer Science, Complutense U. of Madrid, elvira@sip.ucm.es
[3] Depts. of Comp. Sci. and El. and Comp. Eng., U. of New Mexico, herme@unm.edu

The relationship between abstract interpretation [2] and partial evaluation [5] has received considerable attention and (partial) integrations have been proposed starting from both the partial deduction (see e.g. [6] and its references) and abstract interpretation perspectives. Abstract interpretation-based analyzers (such as the `CiaoPP` analyzer [9, 4]) generally compute a *program analysis graph* [1] in order to propagate (abstract) call and success information by performing fixpoint computations when needed. On the other hand, partial deduction methods [7] incorporate powerful techniques for on-line specialization including (concrete) call propagation and unfolding.

In this work we propose what we argue is the first generic framework for the efficient and precise integration of abstract interpretation and partial deduction from an abstract interpretation perspective, and which combines the best of both worlds. As starting point, we consider state-of-the-art algorithms for context-sensitive, polyvariant abstract interpretation [9, 4]. The central idea in this novel framework is to extend such algorithms, which already incorporate success propagation, such that calls which appear dynamically in the analysis graph are not analyzed w.r.t. the definition of the procedure in the original program but w.r.t. possibly *new, specialized definitions* of these procedures. These specialized definitions are obtained by applying powerful techniques for on-line program specialization, including unfolding and abstract executability [10]. Abstract executability allows exploiting analysis information in order to (abstractly) execute certain atoms, which in turn may allow unfolding of other atoms. Also, performing unfolding steps allows us to prune away useless branches, which will result in improved success information. Furthermore, propagating (abstract) success information simultaneously will result in an improved unfolding. Therefore, key ingredients of our proposal include the accurate success propagation inherent to context-sensitive abstract interpretation and the powerful constant propagation and program transformations achievable by partial deduction.

It should be noted that existing proposals for such integration use abstract interpretation as a *means* for improving partial evaluation rather than as a *goal*

at the same level as producing a specialized program. This implies that, as a result, their objective is to yield a set of atoms which determines a partial evaluation rather than to compute a safe approximation of its success. In contrast, a fundamental objective of our work is to improve success information by analyzing the specialized code, rather than the original one. We achieve this objective by smoothly *interleaving* both techniques and this, on one hand, improves success information, even for abstract domains which are not related directly to partial evaluation. On the other hand, with more accurate success information we can improve further the quality of partial evaluation. The overall method thus yields not only a specialized program but also a safe approximation of its behavior.

Our framework is parametric w.r.t. different control strategies (both for local and global control [3]) and abstract domains (including non *downwards-closed* properties). Different combinations of such parameters correspond to existing algorithms for program analysis and specialization. Simultaneously, our approach opens the door to strictly more precise results than those achievable by each of the individual techniques. The framework has been implemented in the context of the `CiaoPP` analysis and specialization system. A complete description of the method (and related techniques) can be found in [8].

References

1. M. Bruynooghe. A Practical Framework for the Abstract Interpretation of Logic Programs. *Journal of Logic Programming*, 10:91–124, 1991.
2. P. Cousot and R. Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. of POPL'77*, pages 238–252, 1977.
3. J.P. Gallagher. Tutorial on specialisation of logic programs. In *Proceedings of PEPM'93, the ACM Sigplan Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 88–98. ACM Press, 1993.
4. M. Hermenegildo, G. Puebla, K. Marriott, and P. Stuckey. Incremental Analysis of Constraint Logic Programs. *ACM Transactions on Programming Languages and Systems*, 22(2):187–223, March 2000.
5. N.D. Jones, C.K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice Hall, New York, 1993.
6. M. Leuschel. A framework for the integration of partial evaluation and abstract interpretation of logic programs. *ACM Transactions on Programming Languages and Systems*, 26(3):413 – 463, May 2004.
7. J. W. Lloyd and J. C. Shepherdson. Partial evaluation in logic programming. *The Journal of Logic Programming*, 11:217–242, 1991.
8. G. Puebla, E. Albert, and M. Hermenegildo. Abstract Interpretation with Specialized Definitions. Technical Report CLIP6/2005.0, Technical University of Madrid, School of Computer Science, UPM, July 2005.
9. G. Puebla and M. Hermenegildo. Optimized Algorithms for the Incremental Analysis of Logic Programs. In *Proc. of SAS'96*, pages 270–284. Springer LNCS 1145, 1996.
10. G. Puebla and M. Hermenegildo. Abstract Multiple Specialization and its Application to Program Parallelization. *J. of Logic Programming.*, 41(2&3):279–316, November 1999.