Design of Syntactic Program Transformations by Abstract Interpretation of Semantic Transformations *

Patrick Cousot

Département d'informatique
École Normale Supérieure
45 rue d'Ulm
75230 Paris cedex 05, France
Patrick.Cousot@ens.fr
http://www.di.ens.fr/~cousot/

Traditionally, static program analysis has been used for offline program transformation i.e. an abstraction of the subject program semantics is used to determine which syntactic transformations are applicable. A classical example is binding-time analysis before partial evaluation [4, 5].

We present a new application of abstract interpretation to the formalization of source to source program transformations:

– The semantic transformation is understood as an abstraction of the subject program semantics. The intuition is that the transformed semantics is an approximation of the subject semantics because, most often, redundant elements of the subject semantics have been eliminated;

– The correctness of the semantic transformation is expressed by an observational abstraction. The intuition is that the subject and transformed semantics should be exactly the same when abstracting away from irrelevant hence unobserved details;

– Finally, the syntax of a program is shown to be an abstraction of its semantics (in that details of the execution are lost) so that the transformed program is an abstraction of the transformed semantics.

Abstract interpretation theory [1, 2] provides the ingredients for designing a syntactic source-to-source transformation as an abstraction of a semantics-to-semantics transformation, which correctness is formally established through an observational abstraction. In particular iterative transformation algorithms are abstraction of the fixpoint semantics of the subject program.

Several examples have been studied with this perspective such as blocking command elimination [3], program reduction, constant propagation, partial evaluation, etc.

* This work was supported in part by the european FP5 project IST-1999-20527 Daedalus.

References

1. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In 6^{th} *POPL*, pages 269–282, San Antonio, TX, 1979. ACM Press.
2. P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, Aug. 1992.
3. P. Cousot and R. Cousot. A case study in abstract interpretation based program transformation: Blocking command elimination. *ENTCS*, 45, 2001. `http://www.elsevier.nl/locate/entcs/volume45.html`, 23 pages.
4. N. Jones, C.K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation.* Int. Series in Computer Science. Prentice-Hall, June 1993.
5. N.D. Jones. An introduction to partial evaluation. *ACM Comput. Surv.*, 28(3):480–504, Sep. 1996.

P. Codognet (Ed.): ICLP 2001, LNCS 2237, pp. 4–5, 2001.