A General Infra-structure for Interprocedural Analysis of Concurrent C

Helmut Seidl
Technische Universität München, Germany

This tutorial summarizes joint work with Kalmer Apinis and Vesal Vojdani.

This tutorial is about infrastructures for general-purpose interprocedural analyses. It consists of two parts. The first part argues that side-effecting constraint systems may serve as kind of a swiss army knife for specifying analyses, while the second part provides an overview on solving techniques for such systems.

Side-effecting constraint systems were originally introduced for the analysis of multi-threaded code by Müller-Olm, Seidl and Vene (2003).

Here, we show how this formalism provides a unified framework for realizing efficient interprocedural analyses of programs, possibly with dynamic function calls, where the amount of context-sensitivity can be tweaked and where the context-sensitive analyses of local properties can be combined with *flow-insensitive* analyses of global properties, e.g., about the heap.

One infrastructure realizing this intermediate format, is the analyzer generator Goblint, which we used to practically evaluate this approach on real-world examples.

The second part reports on techniques for solving side-effecting constraint systems. One major issue here is that non-trivial analysis problems require complete lattices with infinite ascending and descending chains. In order to compute reasonably precise post-fixpoints of the resulting systems of equations, Cousot and Cousot have suggested accelerated fixpoint iteration by means of widening and narrowing [1,2,4].

The strict separation into phases, however, may unnecessarily give up precision that cannot be recovered later. While widening is also applicable if equations are non-monotonic, this is no longer the case for narrowing. A narrowing iteration to improve a given post-fixpoint, additionally, must assume that all right-hand sides are monotonic.

The latter assumption, though, is not met in presence of widening. It is also not met by constraint systems corresponding to context-sensitive interprocedural analysis, possibly combining context-sensitive analysis of local information with flow-insensitive analysis of globals.

As a remedy, we present a novel operator that combines a given widening operator with a given narrowing operator. We present adapted versions of round-robin as well as of worklist iteration, local, and side-effecting solving algorithms for this combined operator and prove that the resulting solvers always return sound results and are guaranteed to terminate for monotonic systems whenever only finitely many unknowns (constraint variables) are encountered.

References

[1] P.Cousot, R.Cousot. *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* POPL'77, pp. 238–252, ACM, 1977.

[2] P.Cousot, R.Cousot. *Static Determination of Dynamic Properties of Recursive Procedures.* IFIP Conf. on Formal Description of Programming Concepts, pp. 237–277, North-Holland, 1977.

[3] M.Sharir, A.Pnueli. *Two Approaches to Interprocedural Data Flow Analysis.* S.S.Muchnick, N.D.Jones (eds.), Program Flow Analysis: Theory and Application, pp. 189–233, Prentice-Hall, 1981.

[4] P.Cousot, R.Cousot. *Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation.* PLILP'92, pp. 269–295, 1992.
http://dx.doi.org/10.1007/3-540-55844-6_101

[5] M.Bruynooghe, M.Wirsing. *Programming Language Implementation and Logic Programming.* PLILP'92, LNCS 631, Springer, 1992.

[6] K.Apinis, H.Seidl, V.Vojdani. *Side-Effecting Constraint Systems: A Swiss Army Knife for Program Analysis.* APLAS'12, pp. 157–172, 2012.
http://dx.doi.org/10.1007/978-3-642-35182-2_12

[7] R.Jhala, A.Igarashi. *Programming Languages and Systems.* APLAS'12, LNCS 7705, Springer, 2012. http://dx.doi.org/10.1007/978-3-642-35182-2

[8] K.Apinis, H.Seidl, V.Vojdani. *How to Combine Widening with Narrowing for Non-monotonic Systems of Equations in Program Analysis.* PLDI'13, 2013 (to appear).