Newtonian Program Analysis

Javier Esparza
Technische Universität München, Germany

Static program analysis is the process of obtaining information about the behaviour of a program without actually executing its code.

From a mathematical point of view, static program analysis proceeds in three steps. First, the code is transformed into a formal system of fixed-point equations

$$\begin{aligned} x_1 &= f_1(x_1, \ldots, x_n) \\ &\cdots \\ x_n &= f_n(x_1, \ldots, x_n) \end{aligned}$$

Then, the equations are interpreted by fixing a domain and the meaning of the functions f_i; these are chosen so that the smallest solution of the equations (i.e., the least fixed-point of the vector $(f_1, \ldots, f_n)$ of functions) contains the information on the program one wants to obtain. Finally, the smallest solution is computed or approximated.

In the course I present generic methods for solving such equations, i.e., methods that are well-defined for any interpretation. After reviewing the classical worklist algorithm derived from the Knaster-Tarski and Kleene theorems, I show that Newton's method – a well-known technique for numerically solving equations with the real numbers as domain – can be extended to arbitrary interpretations. I present consequences and applications for other domains like languages or semilinear sets.

References

The course is based on:

1. J. Esparza, S. Kiefer, M. Luttenberger. *Newtonian Program Analysis*. Technical report, Technische Universität München, 2009.
   Available at `http://www7.in.tum.de/um/bibdb/kiefer/newtProgAn.pdf`

Other papers useful for the course are:

2. P. Cousot, R. Cousot. *Abstract Interpretation Frameworks*. Journal of Logic and Computation 2(4), 1992.
3. J.B Kam, J.D. Ullman. *Monotone Data Flow Analysis Frameworks*. Acta Informatica 7(3), 1977.
4. A. Tarski. *A Lattice-theoretical Fixpoint Theorem and its Applications*. Pacific Journal of Mathematics 5:2, 1955.
   A link to the paper can be found in Wikipedia's page on the Knaster-Tarski theorem.