Randomized Algorithms for Program Analysis and Verification

George C. Necula and Sumit Gulwani

Department of Electrical Engineering and Computer Science,
University of California, Berkeley
{necula, gulwani}@cs.berkeley.edu

Program analysis and verification are provably hard, and we have learned not to expect perfect results. We are accustomed to pay this cost in terms of incompleteness and algorithm complexity. Recently we have started to investigate what benefits we could expect if we are willing to trade off controlled amounts of soundness. This talk describes a number of randomized program analysis algorithms which are simpler, and in many cases have lower computational complexity, than the corresponding deterministic algorithms. The price paid is that such algorithms may, in rare occasions, infer properties that are not true. We describe both the intuitions and the technical arguments that allow us to evaluate and control the probability that an erroneous result is returned, in terms of various parameters of the algorithm. These arguments will also shed light on the limitations of such randomized algorithms.

The randomized algorithms for program analysis are structured in a manner similar to an interpreter. The key insight is that a concrete interpreter is forced to ignore half of the state space at each branching point in a program. Instead, a random interpreter executes both branches of a conditional and combines the resulting states at the join point using a linear combination with random weights. This function has the property that it preserves all linear invariants between program variables, although it may introduce false linear relationships with low probability. This insight leads to a quadratic (in program size) algorithm for inferring linear relationships among program variables, which is both simpler and faster than the cubic deterministic algorithm due to Karr (1976). This strategy can be extended beyond linear equality invariants, to equality modulo uninterpreted functions, a problem called global value numbering. This results in the first polynomial-time algorithm for global value numbering (randomized or deterministic).

These ideas have application in automated deduction as well. We describe a satisfiability procedure for uninterpreted functions and linear arithmetic. Somewhat surprisingly, it is possible to extend the randomized satisfiability procedure to produce satisfying models for the satisfiable problems, and proofs for the unsatisfiable problems. This allows us to detect by proof checking all instances when the randomized algorithm runs unsoundly.

We will also show that it is possible to integrate symbolic and randomized techniques to produce algorithms for more complex problems. We show that in this manner we can extend in a natural way randomized algorithms to interprocedural analyses.