Trace Abstraction

Andreas Podelski

University of Freiburg, Germany

Abstract. Trace abstraction refers to a new approach to program verification algorithms. Instead of trying to construct a proof for the input program directly, we first construct auxiliary programs from proofs. We construct each auxiliary program (which can be of general form) from the proof for a program in a specific form, namely a program in the form of a trace (i.e., a sequence of statements). A trace is a program (where the statements have a semantics). At the same time, a trace is a word over a finite alphabet (where the semantics of statements is ignored). As a word, a sequence of statements can be read by an automaton. Just as we ask whether there exists an accepting run of a given automaton on a sequence of letters, we can ask whether there exists a correctness proof for a sequence of statements, a correctness proof that can be assembled from a given finite set of Hoare triples. We iteratively construct auxiliary programs from proofs for traces. The iteration stops when the constructed programs together cover all possible behaviors of the input program. A crucial step here is the covering check. This step is based on algorithms for automata (inclusion test, minimization, ...). The approach applies to a range of verification problems, for sequential programs with (possibly recursive) procedures and concurrent programs with possibly unboundedly many threads, and even to real-time programs.