Semantics-based Program Verification using K and Matching logic

Grigore Rosu

University of Illinois at Urbana-Champaign

Abstract

One of the long-lasting dreams of the programming language community is to have one formal semantic definition of a target programming language and from it to derive all the tools needed to execute and analyze programs written in that language: parsers, interpreters, compilers, symbolic execution engines, model checkers, deductive program verifiers, and so on. We believe that this is not a dream anymore. In this talk we will give an overview of K, a rewrite-based executable semantic framework in which programming languages can be defined using configurations, computations and rules. Configurations organize the state in units called cells, which are labeled and can be nested. Computations carry computational meaning as special nested list structures sequentializing computational tasks, such as fragments of program. Computations extend the original language abstract syntax. K (rewrite) rules make it explicit which parts of the term they read-only, write-only, read-write, or do not care about. This makes K suitable for defining truly concurrent languages even in the presence of sharing. Computations are like any other terms in a rewriting environment: they can be matched, moved from one place to another, modified, or deleted. This makes K suitable for defining control-intensive features such as abrupt termination, exceptions or call/cc. The K framework is designed to allow implementing a variety of generic tools that can be used with any language defined in K. Currently, it includes the following: a parser, interpreters, symbolic execution engines (connected to the Z3 SMT solver), state-space exploration via a configurable semantic debugger, a model checker, and a deductive program verifier. The latter is based on matching logic for expressing static properties, which generalizes separation logic, and on reachability logic for expressing dynamic properties, which generalizes Hoare logic.