Why and How Does K work? The Logical Infrastructure Behind It

(invited talk abstract)

Grigore Rosu
University of Illinois at Urbana-Champaign
grosu@illinois.edu

The K framework was born from our firm belief that an ideal language framework is possible. Specifically, that programming languages must have formal definitions, and that tools for a given language, such as interpreters, compilers, state-space explorers, model checkers, deductive program verifiers, etc., are derived from just one reference formal definition of the language, correct-by-construction and at no additional cost specific to that language. No other semantics for the same language should be needed. Several real languages have been formalized their semantics in K, including C, Java, JavaScript, PHP, Python, Rust, and more recently the Etherem VM (EVM), and the generic K tools have been instantiated to with these languages. In particular, the EVM semantics is used commercially by Runtime Verification to formally verify smart contracts on the Ethereum blockchain.

But what is behind K? Why and how does it work? This talk will discuss the logical formalism underlying K, matching logic, a first-order logic (FOL) variant for specifying and reasoning about structure by means of patterns and pattern matching. Matching logic generalizes several logical frameworks important for program analysis, such as: propositional logic, algebraic specification, FOL with equality, (polyadic) modal logics, temporal logics, separation logic, as well as dynamic and Hoare logics. Binders and fixed-points can also be defined in matching logic, and thus the variety of lambda/mu-calculi and substitution-based semantics. Patterns can specify both structural requirements, including separation requirements at any level in any program configuration (not only in the "heap"), as well as logical requirements and constraints. The various languages defined in K, regardless of their size and complexity, become nothing but matching logic theories, and the various tools provided by the K framework, such as interpretion, symbolic execution, search and model checking, as well as full-fledged deductive program verification, become nothing but proof search heuristics in matching logic.

Submitted to:
LFMTP 2018