Reg. No: 2019/33/B/ST6/00289; Principal Investigator: prof. dr hab. Witold Charatonik

Abstract Machines for Programming Languages: Investigations in Formal Interderivations

Nowadays we use computer-based systems on a daily basis. Their functioning is based on two important principles that we implicitly assume: first, that the computer program coded by a programmer in a high-level programming language is correct and agrees with its specification; and second, that the computer correctly (and efficiently) executes the instructions encoded by the program.

Programming languages are artificial languages used in communication with computers as a means to precisely describe tasks to perform by the computer. Just as in human communication using natural language, communication with computers cannot be successful unless both sides have a good grasp of the language they use. Programming languages differ from natural languages in that it is essential that the meaning of language constructs be precisely defined so that they can be unambiguously understood. Even small mistakes or ambiguities may have serious consequences, especially in safety-critical systems, where human lives depend on the correctness of both software and hardware.

Therefore each programming language should have a precise, mathematically defined semantics that can be used to formally reason about the language itself and about programs written in it. In programming-languages research many different ways of describing semantics are used, depending on the needs of language designers, programmers, or implementors. Apart from high-level semantic formats used by programmers we need to be able to implement the language on a computer, where a low-level, detailed and executable semantics is needed. Nonetheless, all the various description of the meaning of the language should be equivalent so that it could be proved that the computer executes exactly what is prescribed in the program.

The main goal of this project is to develop new tools and methods—both theoretical and practical—to study functional programming languages, and in particular various formats of their formal semantics. Functional programming languages constitute a group of languages of prime importance in computer science. Some of them (such as ML or Haskell) are used in many commercial applications, even though they are not so popular as, e.g., object-oriented languages. However, as the research progresses, and also in reaction to constant changes of the IT industry needs, more constructions and tools developed in the functional paradigm find their way to other popular languages, such as Scala, Ruby, or Java.

A particular format of semantics, used also for functional languages, is an abstract machine. It is an intermediate format between a high-level semantics easily understood by humans but abstracting from many details, and low-level semantics describing all the particulars of implementation. Abstract machines are used both as a theoretical tool to study properties of programming languages, as well as a prototypical model of implementation where efficiency is an important issue. The specific goal of this project is to interderive various formats of operational semantics for functional languages and facilitate methodical generation of efficient abstract machines from high-level specifications, in a way that preserves their correctness.