The Data-Flow Perspective on Static Single-Assignment Form

Baltasar Trancón Y Widemann and Markus Lepper

[1] TU Ilmenau
[2] semantics GmbH

Abstract. The static single-assignment (SSA) form is a low-level intermediate representation of computer programs. It has been designed to make the analysis and transformation of imperative code easier, by providing just the right amount of referential transparency of local variables. That SSA does its job fairly well is proven by its central use in many current state-of-the-art compilers. But the situation is something of a paradox. The languages that are translated to SSA typically possess many features that quite defy its purpose: Mutable arrays and objects with reference semantics on the one hand; concurrency and non-local control flow on the other. We take a fresh look at the design principles of SSA from the perspective of a domain-specific, semantically rigorous paradigm, namely total functional synchronous data-flow programming, embodied in the prototypic language Sig. We demonstrate that the expressivity of SSA is far more complete and foundational there. The same form has many interpretations, in notably as a data-flow diagram, the IR of a functional program, and both an intensional (Z schema-like) and a propositional definition of element-wise denotational semantics. We also demonstrate how the single operation particular to SSA, the phi node, naturally suggests semantically rigorous solutions to two principal semantical problems of the data-flow approach, namely initialization and control flow. Both are necessary for real-world applications, but notoriously ill-supported in many established practical programming systems.