Formal Component-Based Semantics

Ken Madlener[1], Sjaak Smetsers[1], and Marko van Eekelen[12]

[1] Institute for Computing and Information Sciences (iCIS),
Radboud University Nijmegen, The Netherlands
[2] School of Computer Science, Open University of the Netherlands
{k.madlener,s.smetsers,m.vaneekelen}@cs.ru.nl

Formalized programming language semantics has many useful applications, e.g. formal proofs of specific properties of specific programs, proof certificates for generated code and proving conformance of the compiler to the formal semantics of the language. A widely used framework for the formalization of semantics in theorem provers is Structural Operational Semantics (SOS) [3]. A drawback of SOS is its lack of modularity: auxilliary entities, such as environments and stores, need to be threaded through *all* rules defining the transition relation, preventing the reuse of common language constructs. This is even more so a serious drawback for theorem proving, because it forgoes the modular development of meta-theoretic proofs about the language.

Modular SOS (MSOS), proposed by Peter D. Mosses [1], is a simple variant of SOS that addresses the lack of modularity. It provides a direct way to propagate all the entities that are not mentioned by the rule in question between the premise(s) and conclusion of that rule. MSOS allows the independent description of programming language components, enabling the reuse of significant parts of one language in the description of other languages. One may even go as far as setting up an (open-ended) repository of language constructs that contains every conceivable independent language component [2]. The components of the repository may have small proofs attached to them that may be composed into larger proofs of properties that hold for a full language.

We have developed a way to embed MSOS in the theorem prover Coq [4] that enables the user to express individual components in separate Coq files. This is fully compatible with component-based semantics, and we have already built a small repository of language components using our framework. Our formalization enables modular proof of meta-properties. As an example, we develop a modular proof of determinism for a mini-language.

The formalization follows the original design of MSOS in its use of arrows of a category for labels on the transitions. It makes essential use of dependent types, and also uses the recent addition of first-class type classes to Coq.

References

1. P. D. Mosses. Modular structural operational semantics. *J. Log. Algebr. Program.*, 60-61:195–228, 2004.
2. P. D. Mosses. Component-based semantics. In *Proceedings of the 8th international workshop on Specification and verification of component-based systems*, SAVCBS '09, pages 3–10, New York, NY, USA, 2009. ACM.

3. G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.
4. The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.3*, 2010.