Modular SOS for Control Operators

Peter D. Mosses

Department of Computer Science, Swansea University
p.d.mosses@swansea.ac.uk

Abstract for a talk in honour of Gordon Plotkin

In 1981, I attended the wonderful series of lectures given in Aarhus by Gordon Plotkin on "a simple and direct method for specifying the semantics of programming languages", introducing his structural approach to operational semantics (SOS). The lecture notes, which were widely disseminated and subsequently republished as a journal article [1], have been cited more than 4,000 times, and are still being cited more than 100 times per year.

Regarding modularity, Plotkin wrote "we just hope that if we get the other things in a reasonable state, then current ideas for imposing modularity on specifications will prove useful". In the SOS of a programming language, however, the notation used to specify transitions depends on the semantic entities included in configurations (such as environments and stores), and on whether transitions are labelled (e.g., by process interaction signals). When a language specified in SOS is modified or extended, the entities and labels required may change, necessitating tedious global changes and resulting in complicated-looking rules.

The MSOS framework [2] provides a way of obtaining modularity in SOS. Configurations are required to be syntactic terms and computed values; all semantic entities have to be incorporated in labels on transitions. A crucial feature of MSOS is the notation used for specifying labels in rules: it allows referring to particular entities individually, and to all remaining entities collectively. Moreover, labels are morphisms of a category, and labels on adjacent transitions are required to be composable. The rules specifying a language construct need to mention only those entities inherent to its semantics (e.g., environments for a binding construct, stores for imperative variable assignment, signals for process synchronisation). When a language specified in MSOS is modified or extended, the rules for the unaffected constructs never need changing, and remain as simple as possible.

This talk presents a modular technique for specifying *call/cc* and delimited control operators in small-step SOS. Neil Sculthorpe developed it in MSOS, and tested it using semantic prototyping tools [3]; Paolo Torrini proved the specified semantics equivalent to an existing reduction semantics based on evaluation contexts. The results are published in [4]. To employ the same technique in ordinary SOS, all transitions have to be labelled with two new semantic entities, which need to be propagated between the premise and conclusion of each rule used to specify constructs that are not concerned with control; the SOS rules are otherwise unchanged.

References

[1] G. D. Plotkin. A structural approach to operational semantics. *JLAP*, 60–61:17–139, 2004. Originally published as DAIMI FN-19, Dept. of Computer Science, Aarhus Univ., 1981.

[2] P. D. Mosses. Modular structural operational semantics. *JLAP*, 60–61:195–228, 2004.

[3] M. Churchill, P. D. Mosses, N. Sculthorpe, and P. Torrini. Reusable components of semantic specifications. In *Trans. Aspect-Oriented Software Development XII*, volume 8989 of *LNCS*, pages 132–179. Springer, 2015.

[4] N. Sculthorpe, P. Torrini, and P. D. Mosses. A modular structural operational semantics for delimited continuations. In *Proc. Workshop on Continuations, WoC 2015, London, UK*, volume 212 of *EPTCS*, pages 63–80, 2016.