Arrow-Labelled Transition Systems and Modular SOS

Peter D. Mosses
BRICS & Dept. of Computer Science, Univ. of Aarhus, Denmark
http://www.brics.dk/~pdm

Abstract

In SOS descriptions of programming languages (both small-step and big-step), auxiliary information is often included in configurations, or as extra arguments to transition relations. By moving such information to the labels on transitions, and equipping the labels with a partial composition operation, a greater degree of modularity of descriptions (and of bisimulation theories) can be achieved. The required labels can normally be defined as the arrows of product categories. More general constructions of label categories allow the description of interleaving in big-step SOS.

Let $(\Gamma, T, \mathbb{A}, \longrightarrow)$ be a labelled transition system (LTS) with configurations $\gamma \in \Gamma$, terminal configurations $t \in T \subseteq \Gamma$, labels $\alpha \in \mathbb{A}$, and transition relation $\longrightarrow \subseteq \Gamma \times \mathbb{A} \times \Gamma$. The LTS is called an *arrow-labelled* transition system (ALTS) when the labels are the arrows of a category (also denoted $\mathbb{A}$), providing label composition $\alpha_1 ; \alpha_2$, identity labels $\iota \in \mathbb{I} \subseteq \mathbb{A}$, and objects $o \in |\mathbb{A}|$; let $\alpha : o \to o'$ indicate that α has source o and target o'. (Note that the transition system is not itself a category, in general.) A *computation* of an ALTS is a possibly-infinite sequence of transitions $\gamma_i \xrightarrow{\alpha_i} \gamma_{i+1}$, such that the labels on adjacent transitions are composable. As usual, finite computations are required to finish with terminal configurations. ALTS provides the foundations for the Modular Structural Operational Semantics (MSOS) framework [1]:

MSOS is a variant of the familiar SOS framework, where axioms and inference rules are used to define LTS. In MSOS, the configurations are always *purely* syntactic, and any auxiliary information is contained in the labels on transitions, which are the arrows of a category. The label category can be enriched without any reformulation of the MSOS rules, and without affecting the computations of the ALTS that they define. Thus MSOS provides the key to greater modularity in SOS. An example may help see how this works:

$$\frac{e \xrightarrow{\alpha} e'}{\texttt{if}(e)s_1\,\texttt{else}\,s_2 \xrightarrow{\alpha} \texttt{if}(e')s_1\,\texttt{else}\,s_2} \qquad \begin{array}{l} \texttt{if}(tt)s_1\,\texttt{else}\,s_2 \xrightarrow{\iota} s_1 \\ \texttt{if}(ff)s_1\,\texttt{else}\,s_2 \xrightarrow{\iota} s_2 \end{array}$$

Such rules are typical of small-step MSOS. The label variable α ranges over arbitrary arrows in $\mathbb{A}$, reflecting that a step for e involves exactly the same information (environment, side-effects, etc.) as the corr. step for the enclosing if-statement. In contrast, ι is restricted to the identity arrows in $\mathbb{I}$, ensuring that the simple reduction transitions labelled with it above cannot affect the semantic state. See [1, 2, 3] for complete examples of small-step MSOS.

A big-step MSOS uses explicit label composition to sequence sub-computations:

$$\frac{e \xrightarrow{\alpha'} tt \quad s_1 \xrightarrow{\alpha''} () \quad \alpha = \alpha' ; \alpha''}{\texttt{if}(e)\, s_1\, \texttt{else}\, s_2 \xrightarrow{\alpha} ()} \qquad \frac{e \xrightarrow{\alpha'} ff \quad s_2 \xrightarrow{\alpha''} () \quad \alpha = \alpha' ; \alpha''}{\texttt{if}(e)\, s_1\, \texttt{else}\, s_2 \xrightarrow{\alpha} ()}$$

An ALTS $(\Gamma, T, \mathbb{A}, \longrightarrow)$ can be reduced to an LTS $(\Gamma \times |\mathbb{A}|, T \times |\mathbb{A}|, \mathbb{A}, \longrightarrow^\bullet)$ with essentially the same computations by letting $(\gamma, o) \xrightarrow{\alpha}{}^\bullet (\gamma', o')$ iff $\gamma \xrightarrow{\alpha} \gamma'$ and $\alpha : o \to o'$. (The labels in the LTS still determine their source and target objects, which are also components of configurations; this duplication could be removed.) Configurations γ_1 and γ_2 are said to be *strongly arrow-bisimilar* when there exists a strong bisimulation S for $(\Gamma \times |\mathbb{A}|, T \times |\mathbb{A}|, \mathbb{A}, \longrightarrow^\bullet)$ such that $((\gamma_1, o), (\gamma_2, o)) \in S$ for all $o \in |\mathbb{A}|$. Regarding transitions labelled with identity arrows as unobservable, weak arrow-bisimulation can be defined analogously.

MSOS provides some basic label categories: $\boldsymbol{Discrete}(E)$, the discrete category with objects in E, used for environments; $\boldsymbol{Pairs}(S)$, the category with objects in S and arrows in $S \times S$ (composition being as for binary relations), used for stores; and $\boldsymbol{Monoid}(A, f, \tau)$, the one-object category whose arrows form the monoid (A, f, τ), used for actions. The label category transformer $\boldsymbol{LabTrans}(i, \mathbb{B})$ maps any label category $\mathbb{A}$ to the product category $\mathbb{A} \times \mathbb{B}$, and is used to combine the basic label categories (the index i of the new component allows it to be accessed and replaced independently of the presence of other components). Under mild conditions on the form of side-conditions on rules, using $\boldsymbol{LabTrans}(i, \mathbb{B})$ to add a new component to the label category preserves the computations defined by the rules. It also preserves bisimulation equivalence, since bisimulations can be lifted through such label transformations. Note that MSOS rules often conform to restricted SOS formats, such as tyft/tyxt.

An interesting construction on label categories (not previously published) which is *not* expressible using products is to map $\mathbb{A}$ to $\mathbb{A}^*$, the one-object category whose arrows are *arbitrary* finite sequences of arrows of $\mathbb{A}$. A predicate asserting that one sequence is a "shuffle" of two others then allows a *big-step* MSOS for *interleaving* to be given. The original (finite) computations of a small-step MSOS with labels in $\mathbb{A}$ can be retrieved by restricting the labels on the big-step transitions to composable sequences.

Ongoing work concerns providing further algebraic structure on label categories, and a restricted meta-notation for rules to ensure that computations are always preserved when enriching the label category using $\boldsymbol{LabTrans}$.

Much of the above is reported in detail in [1, 2, 3].

References

[1] P. D. Mosses. Foundations of Modular SOS (extended abstract). In *MFCS'99*, vol. 1672 of *LNCS*, pp. 70–80. Springer-Verlag, 1999. Full version: BRICS-RS-99-54, Dept. of Computer Science, Univ. of Aarhus, `http://www.brics.dk`, 1999.

[2] P. D. Mosses. A modular SOS for Action Notation. BRICS-RS-99-56, Dept. of Computer Science, Univ. of Aarhus, `http://www.brics.dk`, 1999.

[3] P. D. Mosses. A modular SOS for ML concurrency primitives. BRICS-RS-99-57, Dept. of Computer Science, Univ. of Aarhus, `http://www.brics.dk`, 1999.