Structural Operational Semantics

The main definitions

R.J. van Glabbeek
National ICT Australia
and School of Computer Science and Engineering
The University of New South Wales
rvg@cs.stanford.edu

Structural Operational Semantics [6, 7] is one of the main methods for defining the meaning of operators in system description languages like CCS [6]. A system behaviour, or *process*, is represented by a closed term built from a collection of operators, and the behaviour of a process is given by its collection of (outgoing) transitions, each specifying the action the process performs by taking this transition, and the process that results after doing so. For each n-ary operator f in the language, a number of *transition rules* are specified that generate the transitions of a term $f(p_1, \ldots, p_n)$ from the transitions (or the absence thereof) of its arguments $p_1, \ldots, p_n$.

For purposes of representation and verification, several behavioural equivalence relations have been defined on processes, of which the most well-known is *strong bisimulation equivalence* [6], and its variants *weak* and *branching* bisimulation equivalence [6, 5], that feature abstraction from internal actions. In order to allow compositional system verification, such equivalence relations need to be *congruences* for the operators under consideration, meaning that the equivalence class of an n-ary operator f applied to arguments $p_1, \ldots, p_n$ is completely determined by the equivalence classes of these arguments. Although strong bisimulation equivalence is a congruence for the operators of CCS and many other languages found in the literature, weak bisimulation equivalence fails to be a congruence for the *choice* or *alternative composition* operator $+$ of CCS. To bypass this problem one uses the coarsest congruence relation for $+$ that is finer than weak bisimulation equivalence, characterised as *rooted weak bisimulation equivalence* [6, 2], which turns out to be a minor variation of weak bisimulation equivalence, and a congruence for all of CCS and many other languages. Analogously, *rooted branching bisimulation* is the coarsest congruence for CCS and many other languages that is finer than branching bisimulation equivalence [5].

In order to streamline the process of proving that a certain equivalence is a congruence for certain operators, and to guide sensible language definitions, syntactic criteria (*rule formats*) for the transition rules in structural operational semantics have been developed, ensuring that the equivalence is a congruence for any operator specified by rules that meet these criteria. One of these is the *GSOS format* of Bloom, Istrail & Meyer [4], generalising an earlier format by De Simone [8]. When adhering to this format, all processes are computably finitely branching, and strong bisimulation equivalence is a congruence [4]. Bloom [3] defines congruence formats for (rooted) weak and branching bisimulation equivalence by imposing additional restrictions on the GSOS format.

1 Preliminaries

In this paper $V = \{x_1, x_2, \ldots\}$ and *Act* are two sets of *variables* and *actions*.

Definition 1 A *signature* is a collection Σ of *function symbols* $f \notin V$ equipped with a function $ar : \Sigma \to \mathbb{N}$. The set $\mathbb{T}(\Sigma)$ of *terms* over a signature Σ is defined recursively by:

- $V \subseteq \mathbb{T}(\Sigma)$,
- if $f \in \Sigma$ and $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$ then $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

A term $c()$ is abbreviated as c. For $t \in \mathbb{T}(\Sigma)$, $var(t)$ denotes the set of variables that occur in t. $T(\Sigma)$ is the set of closed terms over Σ, i.e. the terms $p \in \mathbb{T}(\Sigma)$ with $var(p) = \emptyset$. A Σ-*substitution* σ is a partial function from V to $\mathbb{T}(\Sigma)$. If σ is a substitution and S is any syntactic object, then $\sigma(S)$ denotes the object obtained from S by replacing, for x in the domain of σ, every occurrence of x in S by $\sigma(x)$. In that case $\sigma(S)$ is called a *substitution instance* of S. A Σ-substitution is *closed* if it is a total function from V to $T(\Sigma)$.

Definition 2 Let Σ be a signature. A *positive* Σ-*literal* is an expression $t \xrightarrow{a} t'$ and a *negative* Σ-*literal* an expression $t \stackrel{a}{\not\longrightarrow}$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in Act$. A *transition rule* over Σ is an expression of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (the *premises* of the rule) and α a positive Σ-literal (the *conclusion*). The left- and right-hand side of α are called the *source* and the *target* of the rule, respectively. A rule $\frac{H}{\alpha}$ with $H = \emptyset$ is also written α. A *transition system specification* (*TSS*), written (Σ, R), consists of a signature Σ and a collection R of transition rules over Σ. A TSS is *positive* if the premises of its rules are positive.

Definition 3 [4] A *GSOS* rule is a transition rule such that

- its source has the form $f(x_1, \ldots, x_{ar(f)})$ with $f \in \Sigma$ and $x_i \in V$,
- the left-hand sides of its premises are variables x_i with $1 \leq i \leq ar(f)$,
- the right-hand sides of its positive premises are variables that that are all distinct, and that do not occur in its source,
- its target only contains variables that also occur in its source or premises.

A *GSOS language*, or TSS in GSOS format, is a TSS whose rules are GSOS rules.

Example 1 The following fragment of CCS has the constant 0, unary operators $a.\_$ for $a \in Act$, binary operators $+$ and $\|$, and the GSOS rules below, one for every $\alpha \in Act$ and $a \in A$. Here $Act = A \,\dot\cup\, \{\tau\}$ and $A = \mathcal{N} \,\dot\cup\, \overline{\mathcal{N}}$ with $\mathcal{N}$ a set of *names* and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$ the set of *co-names*. The function $\bar{\cdot}$ is extended to A by $\overline{\overline{a}} = a$.

$$\frac{x_1 \xrightarrow{\alpha} y_1}{x_1 + x_2 \xrightarrow{\alpha} y_1} \qquad \frac{x_2 \xrightarrow{\alpha} y_2}{x_1 + x_2 \xrightarrow{\alpha} y_2} \qquad a.x_1 \xrightarrow{\alpha} x_1$$

$$\frac{x_1 \xrightarrow{\alpha} y_1}{x_1 \| x_2 \xrightarrow{\alpha} y_1 \| x_2} \qquad \frac{x_2 \xrightarrow{\alpha} y_2}{x_1 \| x_2 \xrightarrow{\alpha} x_1 \| y_2} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{\overline{a}} y_2}{x_1 \| x_2 \xrightarrow{\tau} y_1 \| y_2}$$

Definition 4 A *transition* over a signature Σ is a closed positive Σ-literal. With structural recursion on p one defines when a GSOS language $\mathcal{L}$ generates a transition $p \xrightarrow{a} p'$ (notation $p \xrightarrow{a}_{\mathcal{L}} p'$):

$f(p_1, \ldots, p_n) \xrightarrow{a}_{\mathcal{L}} q$ iff $\mathcal{L}$ has a transition rule $\frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ and there is a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $\sigma(t) = q$, such that $p_i \xrightarrow{c}_{\mathcal{L}} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c}_{\mathcal{L}} r)$ for $(x_i \stackrel{c}{\not\longrightarrow}) \in H$.

Definition 5 Two processes t and u are *weak bisimulation equivalent* or *weakly bisimilar* ($t \leftrightarroweq_w u$) if $t\mathcal{R}u$ for a symmetric binary relation $\mathcal{R}$ on processes (a *weak bisimulation*) satisfying, for $a \in Act$,

$$\text{if } p\mathcal{R}q \text{ and } p \xrightarrow{a} p' \text{ then } \exists q_1, q_2, q' \text{ such that } q \Longrightarrow q_1 \xrightarrow{(a)} q_2 \Longrightarrow q' \wedge p'\mathcal{R}q'. \qquad (*)$$

Here $p \Longrightarrow p'$ abbreviates $p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n = p'$ for some $n \geq 0$, whereas $p \xrightarrow{(a)} p'$ abbreviates $(p \xrightarrow{a} p') \vee (a = \tau \wedge p = p')$.

t and u are *η-bisimilar* ($t \leftrightarroweq_\eta u$) if in (*) one additionally requires $p\mathcal{R}q_1$;
t and u are *delay bisimilar* ($t \leftrightarroweq_d u$) if in (*) one additionally requires $q_2 = q'$;
t and u are *branching bisimilar* ($t \leftrightarroweq_b u$) if in (*) one requires both;
t and u are *strongly bisimilar* ($t \leftrightarroweq u$) if in (*) one simply requires $q \xrightarrow{a} q'$.

Two processes t and u are *rooted weak bisimulation equivalent* ($t \leftrightarroweq_{rw} u$), if they satisfy

$$\begin{array}{l}\text{if } t \xrightarrow{a} t' \text{ then } \exists u_1, u_2, u \text{ such that } u \Longrightarrow u_1 \xrightarrow{a} u_2 \Longrightarrow u' \text{ and } t' \leftrightarroweq_w u', \text{ and}\\ \text{if } u \xrightarrow{a} u' \text{ then } \exists t_1, t_2, t \text{ such that } t \Longrightarrow t_1 \xrightarrow{a} t_2 \Longrightarrow t' \text{ and } t' \leftrightarroweq_w u'.\end{array}$$

They are *rooted η-bisimilar* ($t \leftrightarroweq_{r\eta} u$) if above one additionally requires $u_1 = u$, $t_1 = t$, and $t' \leftrightarroweq_\eta u'$, they are *rooted delay bisimilar* ($t \leftrightarroweq_{rd} u$) if one requires $u_2 = u'$, $t_2 = t'$ and $t' \leftrightarroweq_d u'$, and they are *rooted branching bisimilar* ($t \leftrightarroweq_{rb} u$) if one requires $u_1 = u$, $u_2 = u'$, $t_1 = t$, $t_2 = t'$ and $t' \leftrightarroweq_b u'$.

It is well known and easy to check that the nine relations on processes defined above are equivalence relations indeed [1, 5], and that, for $x \in \{\text{weak}, \eta, \text{delay}, \text{branching}, \text{strong}\}$, x-bisimulation equivalence is the largest x-bisimulation relation on processes. Moreover, $p \leftrightarroweq_{rx} q$ implies $p \leftrightarroweq_x q$.

Definition 6 An equivalence relation $\sim$ on processes is a *congruence* if

$$p_i \sim q_i \text{ for } i = 1, \ldots, ar(f) \quad \Rightarrow \quad f(p_1, \ldots, p_{ar(f)}) \sim f(q_1, \ldots, q_{ar(f)})$$

for all $f \in \Sigma$. This is equivalent to the requirement that for all $t \in \mathbb{T}(\Sigma)$ and closed substitutions $\sigma, \nu : V \to T(\Sigma)$

$$\sigma(x) = \nu(x) \text{ for } x \in var(t) \quad \Rightarrow \quad \sigma(t) = \nu(t).$$

Theorem 1 On any GSOS language, $\leftrightarroweq$ is a congruence.

References

[1] T. Basten. Branching bisimulation is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.

[2] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[3] B. Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science*, 146:25–68, 1995.

[4] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, January 1995.

[5] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[6] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 19, pages 1201–1242. Elsevier Science Publishers B.V. (North-Holland), 1990. Alternatively see *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, 1989, of which an earlier version appeared as *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag, 1980.

[7] G.D. Plotkin. A structural approach to operational semantics. *The Journal of Logic and Algebraic Programming*, 60–61:17–139, 2004. Originally appeared in 1981.

[8] R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.