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(p₁, . . . , pₙ) from the transitions (or the absence thereof) of its arguments p₁, . . . , pₙ.

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 p₁, . . . , pₙ 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 \( \text{Act} \) are two sets of variables and actions.

Definition 1 A signature is a collection \( \Sigma \) of function symbols \( f \not\in V \) equipped with a function \( \text{ar} : \Sigma \to \mathbb{N} \). The set \( \mathcal{T}(\Sigma) \) of terms over a signature \( \Sigma \) is defined recursively by:

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

A term \( c() \) is abbreviated as \( c \). For \( t \in \mathcal{T}(\Sigma) \), \( \text{var}(t) \) denotes the set of variables that occur in \( t \).

The following fragment of CCS has the constant \( 0 \), unary operators \( + \) and \( \| \), and the GSOS rules below, one for every \( \alpha \in \text{Act} \) and \( a \in A \). Here \( \text{Act} = A \cup \{\tau\} \) and \( A = \mathcal{N} \cup \mathcal{N}^\perp \) with \( \mathcal{N} \) a set of names and \( \mathcal{N}^\perp = \{\mathcal{N} \mid a \in \mathcal{N} \} \) the set of co-names. The function \( \tau \) is extended to \( A \) by \( \tau = a \).

\[
\begin{align*}
\frac{x_1 \xrightarrow{\alpha} y_1}{x_1 + x_2 \xrightarrow{\alpha} y_1} & \quad \frac{x_2 \xrightarrow{\alpha} y_2}{x_1 + x_2 \xrightarrow{\alpha} y_2} & \quad \frac{a.x_1 \xrightarrow{\alpha} x_1}{x_1} \\
\frac{x_1 \xrightarrow{\alpha} y_1}{x_1 \parallel x_2 \xrightarrow{\alpha} y_1 \parallel x_2} & \quad \frac{x_2 \xrightarrow{\alpha} y_2}{x_1 \parallel x_2 \xrightarrow{\alpha} y_1 \parallel x_2} & \quad \frac{x_1 \xrightarrow{\alpha} y_1}{x_1 \parallel x_2 \xrightarrow{\tau} y_1 \parallel y_2} \quad \frac{x_2 \xrightarrow{\alpha} y_2}{x_1 \parallel x_2 \xrightarrow{\alpha} y_1 \parallel y_2}
\end{align*}
\]
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 \( L \) generates a transition \( p \xrightarrow{a} p' \) (notation \( p \xrightarrow{a} \_ L \) \( p' \)):

\[
f(p_1, \ldots, p_n) \xrightarrow{a} q \text{ if } L \text{ has a transition rule } \frac{H}{f(x_1, \ldots, x_n)} \xrightarrow{a-t} \text{ and there is a closed substitution } \sigma \text{ with } \sigma(x_i) = p_i \text{ for } i = 1, \ldots, n \text{ and } \sigma(t) = q, \text{ such that } p_i \xrightarrow{c} L \sigma(y) \text{ for } (x_i \xrightarrow{c} y) \in H \text{ and } \neg \exists r(p_i \xrightarrow{c} r) \text{ for } (x_i \xrightarrow{\rho} r) \in H.
\]

Definition 5 Two processes \( t \) and \( u \) are weak bisimulation equivalent or weakly bisimilar \((t \equiv_w u)\) if \( tRu \) for a symmetric binary relation \( R \) on processes (a weak bisimulation) satisfying, for \( a \in Act \),

\[
\text{if } pRq \text{ and } p \xrightarrow{a} p' \text{ then } \exists q_1, q_2, q' \text{ such that } q \Rightarrow q_1 \xrightarrow{(a)} q_2 \Rightarrow q' \wedge p'Rq'. \quad (*)
\]

Here \( p \Rightarrow p' \) 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 \( (p \xrightarrow{a} p') \lor (a = \tau \wedge p = p') \).

\( t \) and \( u \) are η-bisimilar \((t \equiv_{\eta} u)\) if in \( (*) \) one additionally requires \( pRq_1 \);

\( t \) and \( u \) are delay bisimilar \((t \equiv_{d} u)\) if in \( (*) \) one additionally requires \( q_2 = q' \);

\( t \) and \( u \) are branching bisimilar \((t \equiv_{b} u)\) if in \( (*) \) one requires both;

\( t \) and \( u \) are strongly bisimilar \((t \equiv u)\) if in \( (*) \) one simply requires \( q \xrightarrow{a} q' \).

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

\[
\text{if } t \xrightarrow{a} t' \text{ then } \exists u_1, u_2, u \text{ such that } u \Rightarrow u_1 \xrightarrow{a} u_2 \Rightarrow u' \text{ and } t' \equiv_w u', \text{ and if } u \xrightarrow{a} u' \text{ then } \exists t_1, t_2, t \text{ such that } t \Rightarrow t_1 \xrightarrow{a} t_2 \Rightarrow t' \text{ and } t' \equiv_w u'.
\]

They are rooted η-bisimilar \((t \equiv_{r\eta} u)\) if above one additionally requires \( u_1 = u, t_1 = t, \) and \( t' \equiv_{\eta} u' \),

they are rooted delay bisimilar \((t \equiv_{rd} u)\) if one requires \( u_2 = u', t_2 = t' \) and \( t' \equiv_d u' \), and they are rooted branching bisimilar \((t \equiv_{rb} u)\) if one requires \( u_1 = u, u_2 = u', t_1 = t, t_2 = t' \) and \( t' \equiv_b u' \).

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 \( x \in \{\text{weak, } \eta, \text{ delay, branching, strong}\} \), \( x \)-bisimulation equivalence is the largest \( x \)-bisimulation relation on processes. Moreover, \( p \equiv x q \) implies \( p \equiv_x q \).

Definition 6 An equivalence relation \( \sim \) on processes is a congruence if

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

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

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

Theorem 1 On any GSOS language, \( \equiv \) is a congruence.

References


