SOS for Higher Order Processes

(Extended Abstract)

MohammadReza Mousavi[1], Murdoch J. Gabbay[2], and Michel A. Reniers[1]

[1]Department of Computer Science,
Eindhoven University of Technology,
Eindhoven, The Netherlands

[2]Department of Computer Science,
King's College London,
London, UK

Abstract. We lay the foundations for a Structural Operational Semantics (SOS) framework for higher order processes. Then, we propose a number of extensions to Bernstein's *promoted tyft/tyxt* format which aims at proving congruence of strong bisimilarity for higher order processes. The extended format is called *promoted PANTH*. This format is easier to apply and strictly more expressive than the *promoted tyft/tyxt* format. Furthermore, we propose and prove a congruence format for a notion of *higher order bisimilarity* arising naturally from our SOS framework. To illustrate our formats, we apply them to Thomsen's Calculus of Higher Order Communicating Systems (CHOCS).

Key words: Formal Semantics, Structural Operational Semantics, Bisimulation, Congruence, Congruence Rule Formats.

1 Introduction

Bisimilarity, in its different flavors, is a central notion to concurrency theory. Congruence is a very much desired property for bisimilarity which does not generally hold. Congruence is essential for algebraic treatment of bisimilarity as well as for using it in a compositional manner. Thus, it is an interesting question whether a notion of bisimilarity is a congruence for a particular language or not.

This question has been addressed in a great depth and breadth for languages endowed with a Structural Operational Semantics (SOS) [17] (see [1] for an overview). These studies are usually formulated in terms of syntactic formats that induce congruence for a notion of bisimilarity once the SOS rules conform to the formats.

For languages with a higher order notion of behavior (which may emit and receive their own terms as labels), a few proposals exist in the literature [3,14,19]. This work's most direct inspiration is from Bernstein's *promoted tyft/tyxt* format [3] which aims at proving congruence of strong bisimilarity for higher order processes. We lay the foundations for an SOS framework for higher order processes and extend Bernstein's *promoted tyft/tyxt*, making it both easier to use and strictly more expressive.

For processes with a higher order behavior, strong bisimilarity might be too restrictive since it requires the emitted or received processes (shown as labels) to

be syntactically the same. In practice, however, processes are considered important up to their behavior and hence they should be related using a behavioral (and not syntactic) notion of equality. This leads to a *higher order notion of bisimilarity* [2,6,22]. In this paper, we also present and prove a novel format that induces congruence for higher order bisimilarity.

This paper is organized as follows: In the next section, we give more details of our contribution in the context of the literature. Section 3 formally defines our SOS framework and defines the intended notions of bisimilarity and congruence. Based on these concepts, our *promoted PANTH* format is presented in Section 4. Section 5 studies higher order bisimilarity and proposes the *higher order PANTH* format which induces congruence for this notion. We conclude the paper and comment on future work in Section 6.

2 Related Work

From Tyft/tyxt to PANTH. The *tyft/tyxt* format [13] is aimed at proving congruence of strong bisimilarity and it allows for SOS rules of the following forms:

$$\frac{\{t_i \xrightarrow{l_i} y_i \mid i \in I\}}{f(\overrightarrow{x_j}) \xrightarrow{l} t}, \qquad \frac{\{t_i \xrightarrow{l_i} y_i \mid i \in I\}}{x \xrightarrow{l} t},$$

where x_j and y_i are distinct variables ranging over process terms, f is a function symbol or operator (e.g., sequential composition, parallel composition, etc.), I is a (possibly infinite) set of indices and t and t_i's are process terms.

In [12], negative premises of the form $t_i \overset{l_i}{\not\rightarrow}$ were added to the *tyft/tyxt* format, resulting in the *ntyft/ntyxt* format. The format guarantees the congruence property in this extended setting provided that the transition system specification is *stratified*. Stratification is concerned with defining a measure that decreases from the conclusion to negative premises and does not increase from the conclusion to positive premises. Note that the addition of negative premises to the *tyft/tyxt* format is a non-trivial extension in that it increases expressiveness and introduces technical complications with respect to existence and uniqueness of an intended model for the semantics.

Finally, the *PANTH* format [23] (for Predicates And Negative Tyft/tyxt Hybrid format) extends the *ntyft/ntyxt* format. A deduction rule in the *PANTH* format may have predicates, negative predicates, transitions and negative transitions in its premises and a predicate or a transition in its conclusion.

Promoted Tyft/tyxt. Bernstein in [3] proposes the *promoted tyft/tyxt* format which extends the *tyft/tyxt* format by allowing for the use of terms as labels. Rules in this format have the following form:

$$\frac{\{t_i \xrightarrow{t'_i} y_i \mid i \in I\}}{f(\overrightarrow{x_j}) \overset{g(\overrightarrow{z_k})}{\rightarrow} t}, \quad \frac{\{t_i \xrightarrow{t'_i} y_i \mid i \in I\}}{f(\overrightarrow{x_j}) \xrightarrow{z} t}, \quad \frac{\{t_i \xrightarrow{t'_i} y_i \mid i \in I\}}{x \overset{g(\overrightarrow{z_k})}{\rightarrow} t}, \quad \frac{\{t_i \xrightarrow{t'_i} y_i \mid i \in I\}}{x \xrightarrow{z} t}.$$

The intuition behind the symbols in common with the *tyft/tyxt* format remains unchanged. For the rest, g is a function symbol, z_k's and z are variables, variables in the source and label of the conclusion and targets of the premises are all distinct and furthermore, all t'_i's (labels of premises) are assumed to contain at least one function symbol, i.e., they are not variables. Bernstein proves congruence of strong bisimilarity for SOS specifications conforming to the *promoted tyft/tyxt* format.

Promoted PANTH. In this paper, we show that most of the restrictions on labels imposed above are not necessary in general and propose a more general and relaxed format based on the *promoted tyft/tyxt* format of [3]. We call our new format for strong bisimilarity *promoted PANTH*. Furthermore, the *promoted PANTH* format extends syntactic capabilities of the *promoted tyft/tyxt* format by allowing for predicates, negative premises and lists of terms as labels. We show that the *promoted PANTH* format is strictly more expressive than the *promoted tyft/tyxt* format and point out some usual patterns of SOS rules that the *promoted tyft/tyxt* format cannot deal with and the *promoted PANTH* format can.

Proof Methods for Evaluation Systems. The proof method of Howe [14] and related methods such as those proposed in [18] have been used for proving congruence of applicative bisimulation for functional languages. Sangiorgi also proposes a similar framework in [19] for concurrent extensions of lambda-calculi. Although some of the standard concepts of Howe's method, such as abstraction and evaluation structures, are not explicitly present in our framework, as shown by [3], we can still model the systems studied by [14,18,19] and obtain similar results using our formats.

Higher Order Bisimulation and Higher Order PANTH. It was first noted in [2,6] that there is a need for a notion of behavioral equivalence that relates the behavior of labels instead of their syntax. This notion was also used in [21,22] for the Calculus of Higher Order Communicating Systems (CHOCS).

In this paper, we give a general framework for defining the semantics of such systems and proving congruence for the higher order notion of bisimilarity. We also specify CHOCS [22] in our framework, show that the higher order bisimilarity of [22] trivially coincides with ours and conclude that bisimilarity in this framework is indeed a congruence. This way, one can save pages of proof (such as those given explicitly in [22]) for proving congruence.

In [20], it is argued that the higher order notion of bisimilarity may be still too strong for systems with static restriction while it works fine with dynamic restriction of names. It goes beyond the scope of this paper to discuss this issue but the techniques developed here can be useful in formulating congruence metatheorems for other notions of bisimilarity for higher order processes (e.g., normal and context bisimilatities of [20]).

It is worth mentioning that in [3], the *promoted tyft/tyxt* format is used to prove that higher order bisimilarity is a congruence for CHOCS. But to do so,

the semantics of CHOCS is translated into a new semantics and it is shown that higher order bisimilarity in CHOCS coincides with strong bisimilarity in the new semantics. Using our approach, one can save these intermediate steps and arrive at the desired result directly.

Other SOS Frameworks. Our SOS framework is closest to that of [8] (simplified by omitting the binding signatures) for which no known congruence format exists. The *generalized PANTH* format [15] includes variable binding operators (which are not addressed in this paper), but does not allow for terms as labels and hence cannot deal with higher order process algebras such as CHOCS directly. Galpin in [10] defines a multi-sorted SOS framework with terms as labels. However there, the sort of labels is necessarily different from the sort of processes. Thus higher order processes and higher order bisimilarity do not have a natural presentation in the *extended TSS* framework of [10].

3 Preliminaries

3.1 SOS with First Order Labels

Fix an infinite set of **variables** $x, y, \ldots \in V$. A **signature** is a collection of **function symbols** f, g, each with an associated **arity** $ar(f)$ which is the number of arguments of f. We call f a **constant** when $ar(f) = 0$.

(Process) terms $t, t', t_0 \ldots \in \mathcal{T}(\Sigma)$ are inductively defined in the standard way given variables and a signature. Terms $p, q, \ldots \in \mathcal{C}(\Sigma)$ are **closed** when they mention no variables. We tend to write $p, q, p_0, \ldots$ for closed terms. We shall keep Σ fixed but arbitrary henceforth, so we may drop it. Write $\mathcal{L}$ for the set of **finite lists of terms** (of possibly zero length). We write L, L', or (if we want to refer to elements) $\overrightarrow{t_i}$ for lists. Finally, we write $f(\overrightarrow{t_i})$ and by that we mean $f(t_0, \ldots, t_{ar(f)-1})$ by an implicit assumption that the list $\overrightarrow{t_i}$ is of the right length, i.e., $ar(f)$.

A substitution σ replaces variables in a term with other terms. The set of variables appearing in term t is denoted by $vars(t)$. Two substitutions σ and σ' **respect** relation R when for all $x \in V$, $(\sigma(x), \sigma'(x)) \in R$.

A transition system specification, defined below, is a logical way of defining transition relations and predicates on (closed) terms. We need some important basic definitions first:

For a (transition) relation $r \in Rel$ of arity n, $t, t' \in \mathcal{T}$, and $\overrightarrow{t_i} \in \mathcal{L}$ of length n, call $t \xrightarrow{\overrightarrow{t_i}}_r t'$ a **positive** and $t \not\xrightarrow{\overrightarrow{t_i}}_r$ a **negative transition formula**. We call t the **source** of both transitions and t' the **target** of the positive one.

For a predicate $P \in Pred$ of arity n, $t \in \mathcal{T}$, and $\overrightarrow{t_i} \in \mathcal{L}$ of length n, we call $P(\overrightarrow{t_i})\ t$ a **positive predicate formula** and $\neg P(\overrightarrow{t_i})\ t$ a **negative predicate formula**. A (positive or negative) **formula** is a (positive or negative) transition or predicate formula.

We say formulae are **closed** when all the terms they mention are.

A **deduction rule** $dr \in D$ is a tuple (H, c) where H is a set of formulae and c is a positive formula. We call c the **conclusion** and formulae in H **premises**. We write (H, c) as $\frac{H}{c}$.

Definition 1 (Transition System Specification (TSS)) A **transition system specification** is a tuple $(\Sigma, Rel, Pred, D)$ consisting of a signature Σ, disjoint sets of relations Rel and predicates $Pred$ on terms with fixed arities, and a set of deduction rules D.

Note that a transition relation of arity n can be viewed as a predicate of arity $n+1$. [23] also shows how to code predicate formulae as transition formulae with dummy right-hand sides.

In the following example, we give the TSS of a higher order process algebra called CHOCS [22] which serves as a running example throughout the rest of the paper.

Example 1 (Calculus of Higher Order Communicating Systems (CHOCS)) The signature of CHOCS consists of the following operators: 0, a, $\tau.\_$, $c!\_.\_$, $c?a.\_$, $\_ + \_$, $\_ \mid \_$, $\_ \setminus c$ and $\_[S]$ where c is taken from the set C of *channel names*, a from the set A of *atoms* and $S : C \to C$ is a function on channel names. (In [22], atoms are called process variables. To avoid confusion with variables in our SOS setting, we use the term *atom* instead.)

Process 0 is a deadlocking process. An atom a is supposed to represent a "hole" in the process description which can be substituted by another process term. Other than being substituted by a term, an atom does not have any other observable behavior. Internal action prefixing $\tau.p$ first performs a τ-step and then behaves as p. A send prefixed process $c!p.p'$ sends process p along the channel c and becomes p' afterwards. A receive prefixed process $c?a.p$, receives a process along c and substitutes it for atom a in p. Choice is denoted by $+$ and parallel composition by $|$. To make a channel name c internal to process p the restriction expression $p \setminus c$ is used. Finally, renaming expression $p[S]$ renames all channel names of p as specified by the the renaming function S.

The transition relations for this formalism are classes of unary substitution $\xrightarrow{t}_{/a}$, send $\xrightarrow{t}_{c!}$ and receive $\xrightarrow{t}_{c?}$ transitions and a nullary internal action $\rightarrow_\tau$ transition. Substitution transition $p \xrightarrow{p'}_{/a} p''$ stands for "substituting a with p' in p results in p''". Send transition $p \xrightarrow{p'}_{c!} p''$ means that process p emits process p' along channel c and arrives in p'', similarly $p \xrightarrow{p'}_{c?} p''$ means that p receives p' along channel c and becomes p''. No predicates are used in the TSS of CHOCS.

The deduction rules of the CHOCS semantics are given in Figure 1. For brevity, we have omitted the rules dedicated to commutativity of choice and parallel composition. Also, we assume that processes are written in such a way that the substitution happening in the receive rule avoids capture of bound atoms. This can be dealt with explicitly in our SOS framework (cf. [3]) but it will only clutter our presentation and hence we dispense with it.

$$\frac{}{a \xrightarrow{z}_{/a} z} \quad \frac{}{b \xrightarrow{z}_{/a} b} a \neq b \quad \frac{x_0 \xrightarrow{z}_{/a} y_0 \quad x_1 \xrightarrow{z}_{/a} y_1}{c!x_0.x_1 \xrightarrow{z}_{/a} c!y_0.y_1} \quad \frac{x \xrightarrow{z}_{/b} y}{c?a.x \xrightarrow{z}_{/b} c?a.y} a \neq b$$

$$\frac{x_0 \xrightarrow{z}_{/a} y_0 \quad x_1 \xrightarrow{z}_{/a} y_1}{x_0 + x_1 \xrightarrow{z}_{/a} y_0 + y_1} \quad \frac{x_0 \xrightarrow{z}_{/a} y_0 \quad x_1 \xrightarrow{z}_{/a} y_1}{x_0 \mid x_1 \xrightarrow{z}_{/a} y_0 \mid y_1} \quad \frac{x_0 \xrightarrow{z}_{/a} y_0}{x_0 \setminus c \xrightarrow{z}_{/a} y_0 \setminus c} \quad \frac{x_0 \xrightarrow{z}_{/a} y_0}{x_0[S] \xrightarrow{z}_{/a} y_0[S]}$$

$$\frac{}{\tau.x \rightarrow_\tau x} \quad \frac{}{c!x_0.x_1 \xrightarrow{x_0}_{c!} x_1} \quad \frac{x_1 \xrightarrow{z}_{/a} y_1}{c?a.x_1 \xrightarrow{z}_{c?} y_1}$$

$$\frac{x_0 \rightarrow_\tau y_0}{x_0 + x_1 \rightarrow_\tau y_0} \quad \frac{x_0 \xrightarrow{z}_{c!} y_0}{x_0 + x_1 \xrightarrow{z}_{c!} y_0} \quad \frac{x_0 \xrightarrow{z}_{c?} y_0}{x_0 + x_1 \xrightarrow{z}_{c?} y_0}$$

$$\frac{x_0 \rightarrow_\tau y_0}{x_0 \mid x_1 \rightarrow_\tau y_0 \mid x_1} \quad \frac{x_0 \xrightarrow{z}_{c?} y_0 \quad x_1 \xrightarrow{z}_{c!} y_1}{x_0 \mid x_1 \rightarrow_\tau y_0 \mid y_1} \quad \frac{x_0 \xrightarrow{z}_{c!} y_0}{x_0 \mid x_1 \xrightarrow{z}_{c!} y_0 \mid x_1} \quad \frac{x_0 \xrightarrow{z}_{c?} y_0}{x_0 \mid x_1 \xrightarrow{z}_{c?} y_0 \mid x_1}$$

$$\frac{x_0 \rightarrow_\tau y_0}{x_0 \setminus c \rightarrow_\tau y_0 \setminus c} \quad \frac{x_0 \xrightarrow{z}_{c'!} y_0}{x_0 \setminus c \xrightarrow{z}_{c'!} y_0 \setminus c} \; c \neq c' \quad \frac{x_0 \xrightarrow{z}_{c'?} y_0}{x_0 \setminus c \xrightarrow{z}_{c'?} y_0 \setminus c} \; c \neq c'$$

$$\frac{x_0 \rightarrow_\tau y_0}{x_0[S] \rightarrow_\tau y_0[S]} \quad \frac{x_0 \xrightarrow{z}_{c!} y_0}{x_0[S] \xrightarrow{z}_{S(c)!} y_0[S]} \quad \frac{x_0 \xrightarrow{z}_{c?} y_0}{x_0[S] \xrightarrow{z}_{S(c)?} y_0[S]}$$

Fig. 1. Deduction Rules for CHOCS

Not all TSS's induce a unique set of transition relations and predicates. However, in this paper and in all practical cases, it is essential to make sure that a TSS uniquely defines the intended semantics. A criterion that helps in this respect is *stratification* [12] which guarantees that a TSS uniquely defines an intuitive model, called its *stable model*. Since it plays no role in the technical development of this paper, we do not give the details about stratification and only use it in our proofs. Henceforth and without comment, we assume all TSS's under study are stratified and consequently, induce a unique stable model.

Definition 2 (Proof) We say a positive closed formula ϕ is **provable** from a set of positive formulae T and a TSS tss, denoted by $(T, tss) \vdash \phi$ when there is a well-founded upwardly branching tree with nodes labelled by closed formulae such that:

- the root node is labelled by ϕ, and
- if the label of a node q, denoted by ψ, is a positive formula and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above q, then there is a deduction rule $\dfrac{\{\chi_i \mid i \in I\}}{\chi}$ in tss (N.B. χ_i can be a positive or a negative formula) and a substitution σ such that $\sigma(\chi) = \psi$ and for all $i \in I$, $\sigma(\chi_i) = \psi_i$;
- if the label of a node q, denoted by $p \overset{L}{\nrightarrow}$, is a negative formula then there exists no p' such that $p \overset{L}{\rightarrow} p' \in T$ (or similarly, if it is of the form $\neg P(L)p$ then $P(L)p \notin T$).

Definition 3 (Stable Model) A **stable model** defined by tss is a set of positive formulae T such that $\phi \in T$ if and only if $(T, tss) \vdash \phi$, for all closed positive formulae ϕ.

3.2 Bisimilarity

Strong bisimilarity is a natural behavioral equivalence. It is generally too fine-grained (it does not equate enough terms) but it can serve as a basis for other weaker equivalences (e.g., those ignoring internal actions [11]). Congruence formats for weak equivalences (e.g. [4]) are often based on those for strong bisimilarity. Hence, we start with studying strong bisimilarity as an important notion of behavioral equivalence.

We may write pRq for $(p, q) \in R$, or even $\overrightarrow{p_i} R \overrightarrow{q_i}$ to say $\overrightarrow{p_i}$ and $\overrightarrow{q_i}$ have the same length and $p_i R q_i$, for each i.

Definition 4 (Strong Bisimulation and Bisimilarity) Given a TSS (Σ, Rel, $Pred$, D) which induces a unique set of transition relations and predicates, a relation $R \subseteq \mathcal{C} \times \mathcal{C}$ is a **strong simulation** relation if and only if $\forall_{p,q \in \mathcal{C}}\ pRq \Rightarrow$

1. $\forall_{r \in Rel, L \in \mathcal{L}, p' \in \mathcal{C}}\ p \xrightarrow{L}_r p' \Rightarrow \exists_{q' \in \mathcal{C}}\ q \xrightarrow{L}_r q' \wedge p' R q'$;
2. $\forall_{P \in Pred, L \in \mathcal{L}}\ P(L)p \Rightarrow P(L)q$.

A *strong bisimulation* relation is a symmetric strong simulation relation. Closed terms p and q are strongly bisimilar, denoted by $p \leftrightarrows_s q$, if and only if there exists a strong bisimulation relation R such that pRq.

We treat this notion in Section 4 and there, we formulate a congruence meta-theorem for it in Theorem 1.

On one hand, our SOS framework allows for processes as labels. On the other hand processes are usually considered important up to their behavior (and not up to their syntax). Hence, it seems more natural to use a different notion of bisimilarity, rather than the strong one, which not only relates the behavior of source and target processes but also the behavior of label processes. This way, we come to the notion of higher order bisimilarity defined below.

Definition 5 (Higher Order Bisimulation and Bisimilarity [2]) Given a TSS $(\Sigma, Rel, Pred, D)$ which induces a unique set of transition relations and predicates, a relation $R \subseteq \mathcal{C} \times \mathcal{C}$ is a **higher order simulation** relation if and only if $\forall_{p,q \in \mathcal{C}}\ pRq \Rightarrow$

1. $\forall_{r \in Rel, L \in \mathcal{L}, p' \in \mathcal{C}}\ p \xrightarrow{L}_r p' \Rightarrow \exists_{L' \in \mathcal{L}, q' \in \mathcal{C}}\ \ q \xrightarrow{L'}_r q' \wedge LRL' \wedge p' R q'$;
2. $\forall_{P \in Pred, L \in \mathcal{L}}\ P(L)p \Rightarrow \exists_{L' \in \mathcal{L}}\ \ P(L')q \wedge LRL'$.

A *higher order bisimulation* relation is a symmetric higher order simulation relation. Closed terms p and q are higher order bisimilar, denoted by $p \leftrightarrows_h q$, if and only if there exists a higher order bisimulation relation R such that pRq.

We treat this notion in Section 5 and the corresponding congruence results are given in Theorem 3.

Note that higher order bisimilarity is sometimes required to be closed under substitution of atoms [6,22]. Here, we do not add this requirement for the sake of generality but in the coming examples, we show that this additional constraint can easily be coded in the semantic model.

It is also worth noting that higher order bisimilarity, though more natural in our setting, does not make strong bisimilarity obsolete. In some cases, the labels have a syntactic structure and use terms from the language but do not show any behavior, or alternatively, scrutinizing their behavior is a very complex task. In other words, not always terms on the labels are processes or treated as such. In cases, where labels are indeed terms but do not show any observable behavior, all labels are considered equal from a bisimilarity viewpoint and hence higher order bisimilarity renders very weak and impractical. Thus, presenting a meta-theorem for congruence of bisimilarity is interesting even in the presence of terms as labels.

As one might expect, higher order bisimilarity is strictly coarser than strong bisimilarity, i.e., it identifies more processes. Examples of this are shown in the remainder. In Section 5, we also give some sufficient criteria for the two notions to coincide.

3.3 Congruence for Bisimilarity

Next, we define the concept of congruence which is of central importance to our topic.

Definition 6 (Congruence) For a TSS with signature Σ, an equivalence relation $R \subseteq T(\Sigma) \times T(\Sigma)$ is a **congruence** when for all function symbols $f \in \Sigma$ and for all terms $p_i, q_i \in T(\Sigma)$ $(0 \leq i < ar(f))$, if $\overrightarrow{p_i} R \overrightarrow{q_i}$ then $f(\overrightarrow{p_i}) R f(\overrightarrow{q_i})$.

None of the notions of bisimilarity are necessarily a congruence. In the rest of this paper, we endeavor to find sufficient conditions that guarantee them to be a congruence. After all, it turns out that the sufficient conditions for the two notions are somewhat different. A natural question is whether this difference is genuine or not. In the following two examples we show that the notions of congruence for these two equivalences are indeed unrelated, i.e., for neither of the two equivalences, congruence of one implies congruence for the other.

Example 2 $\dfrac{}{f(a) \xrightarrow{a}_r a} \quad \dfrac{}{a \xrightarrow{a}_r a} \quad \dfrac{}{b \xrightarrow{b}_r b}$

Consider the above set of deduction rules defined on the signature a, b and $f(\_)$. In the above TSS, it holds that $a \underline{\leftrightarrow}_h b$ but not $f(a) \underline{\leftrightarrow}_h f(b)$ since $f(a)$ can make an r-transition with label a but $f(b)$ cannot make any transition. Higher order bisimilarity is not a congruence for the above TSS. As for strong bisimilarity, it does not hold that $a \underline{\leftrightarrow}_s b$ in the first place and hence, strong bisimilarity is trivially a congruence.

Example 3 $\dfrac{}{f(a) \xrightarrow{a}_r a} \quad \dfrac{}{f(b) \xrightarrow{b}_r a} \quad \dfrac{}{a \xrightarrow{a}_r a} \quad \dfrac{}{b \xrightarrow{a}_r b}$

Consider the above set of deduction rules defined on the same signature as of Example 2. This time, higher order bisimilarity is a congruence since $a \leftrightarrow_h b$ and $f(a) \leftrightarrow_h f(b)$. However, strong bisimilarity is not a congruence since $a \leftrightarrow_s b$ but not $f(a) \leftrightarrow_s f(b)$.

4 Congruence for Strong Bisimilarity

In this section, we propose a syntactic restriction on TSSs, in the form of a format, that guarantees strong bisimilarity is a congruence. To begin with, we define the auxiliary notion of volatile operators.

4.1 Volatile Operators

Due to the possible interaction between terms and labels, for some operators, it is essential to make sure that transitions with these operators (as labels) are always possible under the change of their arguments by bisimilar ones. First, we give a simple example motivating this concept and then we present the formal definition.

Example 4 $\dfrac{a \xrightarrow{g(x)}_r y}{f(x) \xrightarrow{a}_{r'} y} \quad \dfrac{}{a \xrightarrow{g(a)}_r a} \quad \dfrac{}{b \xrightarrow{g(a)}_r a}$

Consider the above TSS with a and b as constants and f and g as unary function symbols. It holds that $a \leftrightarrow_s b$ but it does not hold that $f(a) \leftrightarrow_s f(b)$ and hence strong bisimilarity is not a congruence.

In this case, we call g *volatile* for r transitions because in the premise of the left-most rule, g appears as a label with an argument that comes from the source of the conclusion of this rule and as such can be replaced by different terms. In order for strong bisimilarity to be a congruence, we require that r-transitions with g in the label should be indifferent to replacing arguments of g by bisimilar ones. However, this is clearly not the case for the middle and rightmost rules since for both an r transition with $g(a)$ is allowed while the same transitions with $g(b)$ are prohibited, thus causing the anomaly.

Definition 7 (Volatile Operators) Given a TSS $(\Sigma, \mathit{Rel}, \mathit{Pred}, D)$ an operator $f \in \Sigma$ is called **volatile** for $r \in \mathit{Rel}$ (similarly for $P \in \mathit{Pred}$) when there exists a rule $d \in D$ of the following form:

$$\frac{\{P_i(L_i)t_i \ \text{ or } \ t_i \xrightarrow{L_i}_{r_i} t'_i \mid i \in I\} \quad \{\neg P_j(L_j)t_j \ \text{ or } \ t_j \not\xrightarrow{L_j}_{r_j} \mid j \in J\}}{P'(L)t \ \text{ or } \ t \xrightarrow{L}_{r'} t'}$$

and $f(\overrightarrow{t_k})$ is a subterm of a component of L_m for some $m \in I \cup J$ such that $r = r_m$ $(P = P_m)$ and $\mathit{vars}(\overrightarrow{t_k}) \cap \mathit{vars}(t) \neq \emptyset$ or $\exists_{i \in I} \mathit{vars}(\overrightarrow{t_k}) \cap \mathit{vars}(t'_i) \neq \emptyset$.

It trivially follows from the above definition that no constant can be volatile.

4.2 Promoted PANTH Format

Next, we formulate our congruence format for strong bisimilarity.

Definition 8 (Promoted PANTH Format) A deduction rule is in the **promoted PANTH** format when it is of the following form

$$\frac{\{P_i(L_i)t_i \;\text{ or }\; t_i \xrightarrow{L_i}_{r_i} y_i \mid i \in I\} \quad \{\neg P_j(L_j)t_j \;\text{ or }\; t_j \overset{L_j}{\nrightarrow}_{r_j} \mid j \in J\}}{P(L)f(\overrightarrow{x_i}) \;\text{ or }\; f(\overrightarrow{x_i}) \xrightarrow{L}_{r} t'}$$

and first, all the variables x_i and y_j ($0 \leq i < ar(f)$ and $j \in I$) and the variables in L are pairwise distinct, second, if a component of L_k ($k \in I \cup J$) is a variable (i.e., does not have any function symbol) then it is not among x_i's and y_j's and third, for all components t of L:

1. if t contains a volatile $g \in \Sigma$ for r (for P) then t is of the form $g(\overrightarrow{z_l})$ where all z_l's are distinct variables and for all $k \in I \cup J$, all components of L_k containing a variable among $\overrightarrow{z_l}$ are of the form $g'(\overrightarrow{t_m})$ where g' is volatile for r_k (for P_k),
2. if there is a volatile operator for r (for P) in the signature and if t is a variable z then for all $k \in I \cup J$, all components of L_k containing z are either z itself or are of the form $g'(\overrightarrow{t_n})$ where g' is volatile for r_k (for P_k).

A TSS is in the *promoted PANTH* format when all its deduction rules are.

Observe that if there is no volatile operator in the signature then none of the two checks on the labels are needed. Volatile operators are very rare in process-algebraic formalisms as it can be observed in the coming examples. Hence, most of the times, the above format can be simplified and checks on the labels can be saved. Surprisingly, the *promoted tyft/tyxt* format is formulated in such a way that all operators can be considered volatile and thus, it turns out to be more restrictive and less expressive than ours. Examples of these phenomena are pointed out next.

Example 5 (Congruence of Strong Bisimilarity for CHOCS) Consider the TSS of CHOCS given in Example 1. No operator in this language is not volatile. All the deduction rules of this TSS are in the *promoted PANTH* format but the one concerning the send operator $c!\_.\_$. This rule violates the format by exploiting variable x_0 in both the source and the label of the conclusion. All the other rules, having a premise are *not* in the *promoted tyft/tyxt* format since they have variables as labels of premises. Note that this restriction of the *promoted tyft/tyxt* format can be seen as a disadvantage since using this format, one cannot deal with ordinary process algebraic operators (e.g., choice and parallel composition) by replacing variables for constant labels. This restriction is not present in the *promoted PANTH* format.

Hitherto, one can imagine two scenarios. Either our format is too weak to capture the congruence of strong bisimilarity for CHOCS (since syntactic formats only give sufficient and not necessary conditions) or strong bisimilarity for

CHOCS is not a congruence in the first place. Fortunately, the latter is the case and this can be shown by a very simple example.

Consider two processes 0 and 0+0. It clearly holds that $0 \underline{\leftrightarrow}_s 0+0$ and $0 \underline{\leftrightarrow}_s 0$ but it does not hold that $c!0.0$ is bisimilar to $c!(0+0).0$ as the former can only perform a $\stackrel{0}{\rightarrow}_{c!}$ transition but the latter can only make a $\stackrel{0+0}{\rightarrow}_{c!}$ a transition and 0 and $0+0$ are not (syntactically) the same terms.

However, one can change the language a bit so that strong bisimilarity becomes a congruence. One such approach is presented in [3] and with a proof of more than a page, it is shown that strong bisimilarity in the new language coincides with a notion of higher order bisimilarity [22] in the original semantics and hence, it is concluded that this notion of higher order bisimilarity for the original language is a congruence. In Section 5, we propose a congruence format for higher order bisimilarity and using that we give a direct proof for congruence of higher order bisimilarity. So, we do not take the approach of [3] in this section.

Alternatively, in order to make the strong bisimilarity a congruence, we propose to change the send operator as follows. First, we change the syntax of a send operator to be a class of unary send operators $c!p._-$ for given closed terms $p \in P$. Then, we change the semantics of the send operator and replace it with this rule: $\dfrac{}{c!p.x_0 \stackrel{p}{\rightarrow}_{c!} x_0}$.

Note that in the above rule the p in the source of the conclusion is part of the function symbol while the p in the label is a term. To check that this rule fits in the *promoted PANTH* format one has to check the following two conditions: first, the set of variables appearing in p and $c!p.x_0$ should be disjoint which holds trivially since the former p is a closed term and second, either p contains no volatile operator or it is of the form $g(\overrightarrow{x})$ for a volatile g. Since the language contains no volatile operator the second obligation is also discharged and hence, we can conclude that strong bisimilarity is a congruence for this slightly modified language. Note that one cannot get a similar result by using the *promoted tyft/tyxt* format for it only allows for labels of the form x or $g(\overrightarrow{x})$ in the conclusion.

Next, by a simple and abstract example, we show that our format is strictly more expressive than the *promoted tyft/tyxt* format of [3].

Example 6 $\dfrac{x \stackrel{z}{\rightarrow}_r y}{f(x) \stackrel{z}{\rightarrow}_r y} \quad \dfrac{}{a \stackrel{f(a)}{\rightarrow}_r b} \quad \dfrac{}{b \stackrel{f(a)}{\rightarrow}_r b}$

Consider a TSS defined by signature $\{a, b, f(\_)\}$, a unary transition relation $\rightarrow_r$, no predicate and the deduction rules given above. None of the three deduction rules are in the *promoted tyft/tyxt* format while they are all in the *promoted PANTH* format and one can check that strong bisimilarity is indeed a congruence. Our claim is that there exists no TSS in the *promoted tyft/tyxt* format that induces the same transition relation as the one induced by the above TSS.

The proof of our claim is quite simple and follows from the proof of Theorem 2.1 in [3]. There, it is shown that, for a TSS in the *promoted tyft/tyxt* format, for all terms $f(\overrightarrow{p_i})$ and $g(\overrightarrow{q_j})$ if there exists $p' \in \mathcal{C}$ and $\overrightarrow{p'_i}, \overrightarrow{q'_j} \in \mathcal{L}$ such

that $f(\overrightarrow{p_i}) \overset{g(\overrightarrow{q_j})}{\rightarrow}_r p'$, $\overrightarrow{p_i} \underline{\leftrightarrow}_s \overrightarrow{p'_i}$ and $\overrightarrow{q_j} \underline{\leftrightarrow}_s \overrightarrow{q'_j}$ then there exists a $p'' \in \mathcal{C}$ such that $f(\overrightarrow{p'_i}) \overset{g(\overrightarrow{q'_j})}{\rightarrow}_r p''$. Getting back to our example, suppose that there exists a TSS in the *promoted tyft/tyxt* format that induces the same transition relation as the one induced by the above TSS. Then, since $a \underline{\leftrightarrow}_s b$ and $f(a) \overset{f(a)}{\rightarrow}_r b$, it should hold that $f(b) \overset{f(b)}{\rightarrow}_r p''$ for some $p'' \in \mathcal{C}$ such that $b \underline{\leftrightarrow}_s p''$. But note that in the transition relation induced by the above TSS, no transition with label $f(b)$ is provable. Q.E.D.

4.3 Characteristic Theorem

Common to [3], we impose an extra constraint on the *promoted PANTH* format to prove congruence, namely the well-foundedness of the TSS under consideration.

Definition 9 (P-Well-Foundedness) For a deduction rule, the **p-variable ordering** $\leq_p$ is an ordering among variables. We wrie $x \leq_p y$, for two variables x and y, when x appears in the source or the label of a premise of the deduction rule and y in the target of the same premise. A TSS is called **p-well-founded** when for all deduction rules in TSS, there is no infinite backward chain of variables with respect to $\leq_p$.

Note that in [7] it has been shown that the well-foundedness assumption, although being very convenient for proofs, is not essential for the *PANTH* format. Indeed, for each non-well-founded TSS in the *PANTH* format, one can construct a well-founded one in a subset of this format (called NTree rules format) that induces the same transition relations and predicates. We leave it open whether the results of [7] carries over to our settings or not.

Theorem 1 (Congruence for Promoted PANTH) For a p-well-founded TSS in the *promoted PANTH* format, strong bisimilarity is a congruence.

5 Congruence for Higher Order Bisimilarity

5.1 Persistency

In this section, we seek sufficient syntactic criteria for the higher order bisimilarity induced by a TSS to be a congruence.

We begin with an auxiliary definition that has the same spirit as that for volatile operators. It is supposed to capture that the labels of a transition can be replaced by bisimilar ones.

Definition 10 Consider a TSS $(\Sigma, Rel, Pred, D)$ and a set Ps of tuples (U, L) where $U \in Rel \cup Pred$ and $L \in \mathcal{L}$. We call Ps a **persistent set** when for all

$(U, L) \in Ps$ and all deduction rules $d \in D$ if d has U in its conclusion then it is of the following form:

$$\frac{\{P(L_i)t_i \ \text{ or } \ t_i \xrightarrow{L_i}_{r_i} y_i \mid i \in I\} \quad \{\neg P(L_j)t_j \ \text{ or } \ t_j \overset{L_j}{\nrightarrow}_{r_j} \mid j \in J\}}{U(L')f(\overrightarrow{x}) \ \text{ or } \ f(\overrightarrow{x}) \xrightarrow{L'}_{U} t'}$$

where $L = \sigma(L')$ for some substitution σ and

1. all x_i's, y_j's ($0 \leq i < ar(f)$ and $j \in I$) and variables appearing in L' are pairwise distinct;
2. for all $k \in I \cup J$, $(r_k, \sigma(L_k)) \in Ps$ (or $(P_k, \sigma(L_k)) \in Ps$).

If a set Ps is persistent and $(U, L) \in Ps$ then we say that **U-transitions (predicates) are persistent for L labels**. A transition relation (predicate) is **persistent** if it is persistent for a label of the form $\overrightarrow{z_i}$ where z_i are distinct variables.

The following theorem gives an idea about the intuition behind persistency.

Theorem 2 If for a TSS all its transition relations and predicates are persistent then:

1. higher order bisimilarity is a congruence;
2. higher order and strong bisimilarity coincide.

Example 7 (Persistency for CHOCS) Substitution, receive and τ-transitions are all persistent in CHOCS, i.e., substitution and receive are persistent for a variable.

5.2 Higher Order PANTH Format

Our criteria are formulated as a syntactic format which we call *higher order PANTH*.

Definition 11 (Higher Order PANTH Format) A deduction rule is in the **higher order PANTH** format when it is of the following form

$$\frac{\{P(L_i)t_i \ \text{ or } \ t_i \xrightarrow{L_i}_{r_i} y_i \mid i \in I\} \quad \{\neg P(L_j)t_j \ \text{ or } \ t_j \overset{L_j}{\nrightarrow}_{r_j} \mid j \in J\}}{P(L)f(\overrightarrow{x_i}) \ \text{ or } \ f(\overrightarrow{x_i}) \xrightarrow{L}_{r} t'}$$

where variables x_i's and y_j's ($0 \leq i < ar(f)$ and $j \in I$) are all pairwise distinct and for all $k \in I \cup J$

1. r_k-transitions (predicates) are persistent for L_k labels (Definition 10);
2. or alternatively, $k \in I$, L_k is a list of distinct variables $\overrightarrow{z_{km}}$ that are distinct from labels of other non-persistent transitions and predicates and are different from x_i's and y_j's.

A TSS is in the *higher order PANTH* format when all its rules are.

Next, we define the notion of well-foundedness for TSS's in the *higher order PANTH* format.

Definition 12 (H-Well-Foundedness) An **h-variable ordering** $\leq_h$ with respect to a deduction rule is an ordering on variables. For two variables x and y, $x \leq_h y$ if x appears in the source of a premise of the rule and y appears in its label or target. A TSS is **h-well-founded** when for all deduction rules in TSS, there is no infinite backward chain of variables with respect to $\leq_h$.

We believe that well-foundedness for this format is a convenience for our proofs and is not a necessary ingredient for congruence but this remains to be formally checked.

Theorem 3 (Congruence for Higher Order PANTH) For an h-well-founded TSS in the *higher order PANTH* format, higher order bisimilarity is a congruence.

Example 8 (Congruence of Higher Order Bisimilarity for CHOCS) The semantics of CHOCS as given in Example 1 conforms to our format. To verify this claim we have to check that in the conclusion of each deduction rule mentions only one function symbol, the targets of premises mention distinct variables and the label of premises either mention distinct variables or are persistent. The first two checks are straightforward. For the third, the only problem arises from the rules having two premises mentioning the same label z. Two of such rules appear in the definition of substitution transitions which is shown to be persistent, so they conform to our format. The only other rule having the same condition is the one defining communication for parallel composition. But in that rule, the receive transition is persistent and hence, the only non-persistent premise (the send transition) trivially satisfies the second criterion of Definition 11. Note that the notion of higher order bisimilarity in [22] also requires that bisimilarity should be closed under substitution of atoms. Our notion does not require this in general, but in the case of CHOCS semantics, the addition of substitution, makes sure that bisimilar terms always have the same "substitution behavior". Hence, the two notions trivially coincide.

6 Conclusion

In this paper, we presented two syntactic formats that guarantee congruence for two notions of strong and higher order bisimilarity. We applied these formats to the CHOCS process algebra [22].

Due to the abundant presence of notions of names and binders in the formalisms with higher order behavior, the addition of these notions to our formats is a very natural and useful extension. We are currently considering this extension and we try to exploit the Gabbay-Pitts Nominal Techniques [9,16] for this purpose.

References

1. L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, 2001.
2. E. Astesiano, A. Giovini, and G. Reggio, Generalized bisimulation in relational specifications. In *Proc. of STACS'88*, volume 294 of *LNCS*, pages 207–226, Springer, 1988.
3. K. L. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *Proc. of LICS'98*, pages 153–164. IEEE CS, 1998.
4. B. Bloom. Structural operational semantics for weak bisimulations. *TCS*, 146:25–68, 1995.
5. R. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *JACM*, 43(5):863–914, 1996.
6. G. Boudol. Towards a lambda-calculus for concurrent and communicating systems. In *Proc. of TAPSOFT'89*, volume 351 of *LNCS*, pages 149–161, Springer, 1989.
7. W. J. Fokkink and R. J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *I&C*, 126(1):1–10, 1996.
8. W. J. Fokkink and C. Verhoef. A conservative look at operational semantics with variable binding. *I&C*, 146(1):24–54, 1998.
9. M. J. Gabbay and J. Cheney. A Sequent Calculus for Nominal Logic, In *Proc. of LICS'04*, pages 139–148, IEEE CS, 2004.
10. V. Galpin. A format for semantic equivalence comparison, *TCS*, 309(1-3):65–109, 2003.
11. R. J. van Glabbeek and W. P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *JACM*, 43(3):555–600, 1996.
12. J. F. Groote. Transition system specifications with negative premises. *TCS*, 118(2):263–299, 1993.
13. J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *I&C*, 100(2):202–260, 1992.
14. D. J. Howe. Proving congruence of bisimulation in functional programming languages. I&C, 124:103–112, 1996.
15. C. A. Middelburg. Variable binding operators in transition system specifications. *JLAP*, 47(1):15–45, 2001.
16. A. M. Pitts. Nominal logic, a first order theory of names and binding. *I&C*, 186(2):165–193, 2003.
17. G. D. Plotkin. A structural approach to operational semantics. *JLAP*, 60:17–139, 2004.
18. D. Sands. From SOS rules to proof principles: An operational metatheory for functional languages. *Proc. of POPL'97*, pages 428-441, ACM Press, 1997.
19. D. Sangiorgi. The Lazy lambda calculus in a concurrency scenario. *I&C*, 111(1):120–153, 1994.
20. D. Sangiorgi. Bisimulation for Higher-Order Process Calculi. *I&C*, 131(2):141–178, 1996.
21. B. Thomsen. Plain CHOCS a second generation calculus for higher order processes. *Acta Informatica*, 30(1):1–59, 1993.
22. B. Thomsen. A theory of higher order communicating systems. *I&C*, 116:38–57, 1995.
23. C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.