A Pre-congruence Format for XY-simulation

Harsh Beohar[1] and Mohammad Reza Mousavi[1]

Center for Research on Embedded Systems
Halmstad University, Sweden
{harsh.beohar,m.r.mousavi}@hh.se

Abstract. XY-simulation is a generalization of bisimulation that is parameterized with two subsets of actions. XY-simulation is known in the literature under different names such as modal refinement, partial bisimulation, and alternating simulation. In this paper, we propose a pre-congruence rule format for XY-simulation. The format allows for checking compositionality of XY-simulation for an arbitrary language with structural operational semantics, by performing very simple checks on the syntactic shape of the rules. We apply our format to derive concrete compositionality results for different notions of behavioral pre-order with respect to different process calculi in the literature.

1 Introduction

XY-simulation is a generalization of bisimulation that is parameterized by two subsets of actions: X and Y [1]. The idea is to weaken the transfer property of a bisimulation relation in the following way: the actions in X are simulated from left to right, while the actions in Y are simulated from right to left. XY-simulation is well-known in the literature, albeit under different names, such as modal refinement [16], partial bisimulation [6], and alternating simulation [4].

An essential property for any notion of behavioral pre-order and hence, also for XY-simulation, is the so-called pre-congruence property. This property allows for compositional verification and reasoning about processes under arbitrary contexts. The pre-congruence property has been studied in the literature for some instances of XY-simulation and for a fixed set of well-known operators from the field of process algebras (see [6, 16] for instance). In this paper, we generalize these results by providing generic sufficient conditions for compositionality of XY-simulation with respect to any arbitrary set of operators with a Structural Operational Semantics (SOS) [21]. We do so by restricting the syntactic shape of the SOS rules to ensure pre-congruence. The result of this paper provides a unified account of existing results and is instantiated to generate new results. Furthermore, the proposed rule format can serve as a yardstick for language designers to check the compositionality of their operators while defining their semantics.

To develop our rule format, we employ the modal decomposition approach proposed in [9, 13] in combination with an existing modal characterization of XY-simulation, due to [11]. We devise a modal decomposition that specifies

when an open term satisfies a modal formula in terms of the modal formulae that are to be satisfied by its variables. This modal decomposition is then directly employed in generating a pre-congruence rule format for XY-simulation. The obtained format is an elegant and simple one; the only specific checks required are simple checks on the labels of the transition formulae, with respect to their inclusion in X or Y. As we demonstrate by some examples in this paper, the format is applicable to various notions of behavioral pre-order and to various process calculi in the literature.

The rest of this paper is structured as follows. In Section 2, we recall the basic definitions that will be used throughout the paper. Then, in Section 3, we first formulate and prove the modal decomposition theorem and using that, derive our pre-congruence rule format. In Section 4, we apply the obtained rule format to various examples from the literature. In Section 5, we show that the syntactic conditions on the rule format cannot be trivially relaxed. Finally, in Section 6, we conclude the paper and present the direction of our ongoing research.

2 Preliminaries

In this section, we first quote the basic definition of labeled transition systems and XY-simulation and some of their properties. Subsequently, we recall a formalization of SOS, and building upon this formalization, we define the basic rule formats that will form the foundations of our results in this paper.

2.1 Transition Systems and XY-simulation

We start by recalling below the well-known notion of labeled transition systems.

Definition 1 (Labeled Transition Systems). *A* labeled transition system (LTS) *is a triple* $(\mathbb{P}, \mathcal{A}, \rightarrow)$*, where* $\mathbb{P}$ *is the set of* processes*,* $\mathcal{A}$ *is the set of* actions*, and* $\rightarrow \subseteq \mathbb{P} \times \mathcal{A} \times \mathbb{P}$ *is the* transition relation*. We denote* $(p, a, q) \in \rightarrow$ *by* $p \xrightarrow{a} q$*.*

The following definition formalizes the notion of XY-simulation, originally due to [1].

Definition 2 (XY-simulation). *Let* $X, Y \subseteq \mathcal{A}$*. A binary relation* $\mathcal{R} \subseteq \mathbb{P} \times \mathbb{P}$ *is an* XY-simulation *relation iff the following transfer conditions are satisfied:*

1. $\forall_{p,a,q,p'}\ (p \xrightarrow{a} p' \wedge p\mathcal{R}q \wedge a \in X) \ \Rightarrow\ \exists_{q'}\ q \xrightarrow{a} q' \wedge p'\mathcal{R}q'$.
2. $\forall_{p,a,q,q'}\ (q \xrightarrow{a} q' \wedge p\mathcal{R}q \wedge a \in Y) \ \Rightarrow\ \exists_{p'}\ p \xrightarrow{a} p' \wedge p'\mathcal{R}q'$.

Two processes $p, q \in \mathbb{P}$ *are* XY-similar*, denoted by* $p \preceq_{X,Y} q$*, iff there is an* XY*-simulation relation* $\mathcal{R}$ *such that* $p\mathcal{R}q$*.*

It is worth noting that in [2], XY-simulation relations are called *covariant-contravariant simulation* relations.

The following lemma lists some of the intuitive properties of XY-similarity.

Lemma 1. *Consider an arbitrary LTS* $(\mathbb{P}, \mathcal{A}, \rightarrow)$ *and assume that* $X, Y, X', Y' \subseteq \mathcal{A}$*; the following statements hold.*

1. *Relation* $\preceq_{X,Y}$ *is a pre-order.*
2. *If* $X \subseteq X'$*, then* $\preceq_{X',Y} \subseteq \preceq_{X,Y}$*.*
3. *If* $Y \subseteq Y'$*, then* $\preceq_{X,Y'} \subseteq \preceq_{X,Y}$*.*
4. $\preceq_{Y,X} = \preceq_{X,Y}^{-1}$*.*

Proof. **1.** It is straightforward to verify that the identity relation is an XY-simulation relation. To prove transitivity, let $p \preceq_{X,Y} p'$ and $p' \preceq_{X,Y} p''$ with $\mathcal{R}$ and $\mathcal{R}'$ their witnessing XY-simulation relations, respectively. It remains to show that $\mathcal{R} \circ \mathcal{R}' = \{(p, p'') \mid \exists_{p'}\ p\mathcal{R}p' \wedge p'\mathcal{R}'p''\}$ is an XY-simulation relation. We distinguish the following cases:

- Let $p \xrightarrow{a} q$, for some $a \in X$, and $p\mathcal{R} \circ \mathcal{R}'p''$. By the definition of relation composition, there exists some p' such that $p\mathcal{R}p'$ and $p'\mathcal{R}'p''$. Since $\mathcal{R}$ and $\mathcal{R}'$ are XY-simulation relations, we have $p' \xrightarrow{a} q'$, $p'' \xrightarrow{a} q''$, and $q\mathcal{R} \circ \mathcal{R}'q''$, for some q', q''.
- Let $p'' \xrightarrow{a} q''$, for some $a \in Y$, and $p\mathcal{R} \circ \mathcal{R}'p''$. Similar to the previous case.

The proof of Items **2.**, **3.**, and **4.** are straightforward from Definition 2. □

Definition 3 (Modal Characterization of XY-simulation). *Let* $\Phi_{X,Y}$ *be the set of modal formulas generated by the following grammar:*

$$\Phi_{X,Y} ::= \bigwedge_{i \in I} \varphi_i \ \mid\ \bigvee_{i \in I} \varphi_i \ \mid\ \langle a \rangle \varphi \ \mid\ [b]\varphi \qquad (a \in X, b \in Y).$$

The semantics of a formula $\varphi \in \Phi_{X,Y}$ *is inductively defined in the standard way, i.e.,*

$$p \models \bigwedge_{i \in I} \varphi_i \iff \forall_{i \in I}\ p \models \varphi_i \qquad p \models \bigvee_{i \in I} \varphi_i \iff \exists_{i \in I}\ p \models \varphi_i$$
$$p \models \langle a \rangle \varphi \iff \exists_q\ p \xrightarrow{a} q \wedge q \models \varphi \qquad p \models [a]\varphi \iff \forall_q\ p \xrightarrow{a} q \Rightarrow q \models \varphi\ .$$

Note that $\top = \bigwedge_{\emptyset}$ and $\bot = \bigvee_{\emptyset}$. Furthermore, we let $\Phi = \Phi_{\mathcal{A},\mathcal{A}}$ and $\varphi_1 \vee \varphi_2 = \bigvee_{i \in \{1,2\}} \varphi_i$. For any two formulas $\varphi, \varphi' \in \Phi$, we define $\varphi \Rightarrow \varphi' = neg(\varphi) \vee \varphi'$, where $neg : \Phi \rightarrow \Phi$ is a function that encodes negation in the logic, by pushing negation through conjunction, disjunction, and the modalities in the standard way.

Theorem 1. $p \preceq_{X,Y} q \iff \forall_{\varphi \in \Phi_{X,Y}}\ p \models \varphi \Rightarrow q \models \varphi$.

Proof. Standard (see [11]). □

2.2 Transition System Specifications

In this section, we recall some basic concepts that are used in the meta-theory of SOS. Regarding the notions treated in this section and the next one, we refer to [3, 19] for more details, examples and results.

Definition 4 (Terms and Signatures). *Let $\mathbb{V}$ be an infinite set of variables with $|\mathbb{V}| \geq |\mathcal{A}|$. A signature is a collection Σ of function symbols $f \notin \mathbb{V}$ equipped with a function $ar : \Sigma \to \mathbb{N}$ denoting their arity. The set $\mathbb{T}(\Sigma)$ of terms over signature Σ is defined as follows:*

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

A *constant* term $c()$ is denoted by c. Let $\mathbf{var}(t)$ denote the set of variables that occur in term t. Let $T(\Sigma) = \{t \mid \mathbf{var}(t) = \emptyset\}$ denote the set of *closed* terms. A *(closed) Σ-substitution* σ is a total function from the set of variables $\mathbb{V}$ to (closed) terms ($T(\Sigma)$) $\mathbb{T}(\Sigma)$.

Definition 5 (Transition System Specifications). *Let Σ be a signature. A* positive Σ-literal *is an expression of the form $t \xrightarrow{a} t'$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in \mathcal{A}$. A* negative Σ-literal *is an expression of the form $t \xnrightarrow{a}$ with $t \in \mathbb{T}(\Sigma)$ and $a \in \mathcal{A}$. A* transition rule *(or simply a rule) over Σ is an expression of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (whose elements are called the* premises *of the rule) and α a Σ-literal (called the* conclusion *of the rule). Furthermore, the left- and the right-hand side (if any) of the conclusion of a rule are called the* source *and the* target *of the rule, respectively. A* transition system specification *(TSS) over Σ is a collection of rules over Σ. A TSS is* standard *if all its rules have positive conclusions and* positive *if moreover all premises of its rules are also positive.*

For each literal α of the form $t \xrightarrow{a} t'$ ($t \xnrightarrow{a}$), the action label of α, denoted by $action(\alpha)$, is defined to be a. For each two terms t, t', literals $t \xrightarrow{a} t'$ and $t \xnrightarrow{a}$ deny *each other.*

A TSS is meant to define an LTS; however, in the presence of negative literals, this is not straightforward. To start with, we first recall the definition of irredundant proof, by Bloom et al. [9], which corresponds to the intuitive notion of proof from a given set of hypotheses.

Definition 6 (Irredundant Proof). *Let P be a TSS over a signature Σ. An* irredundant proof *of a transition rule $\frac{H}{\alpha}$ from P is a well-founded, upwardly branching tree with the nodes labeled by Σ-literals, and some of the leaves marked as "hypotheses", such that:*

- *the root is labeled by α.*
- *H is the set of labels of the hypotheses, and*
- *if β is the label of a node $\star$ which is not a hypothesis and K is the set of labels of the nodes directly above $\star$, then there is a transition rule $\frac{K'}{\beta'}$ in P and substitution σ such that $\sigma(K') = K$ and $\sigma(\beta') = \beta$.*

A proof *of* $\frac{K}{\alpha}$ *from* P *is an irredundant proof of* $\frac{H}{\alpha}$ *from* P *with* $H \subseteq K$.

Note that the term "irredundant" highlights that the set of literals marked as hypotheses in the proof corresponds exactly to the set of premises of the proven rule. In other words, irredundantly provable rules contain no junk literals (i.e., literals not used in the proof tree) among their premises.

Next, we use the notion of irredundant proof to define the LTS associated with a TSS. This is achieved through the following notion of well-supported proof [23].

Definition 7. *Let* P *be a standard TSS over a signature* Σ. *A* well-supported proof *of a closed literal* α *from* P *is a well-founded, upwardly branching tree with the nodes labeled by closed* Σ*-literals, such that the root is labeled by* α *and if* β *is the label of a node* $\star$ *and* K *is the set of labels of the nodes directly above* $\star$, *then*

- *either there is a rule* $\frac{K'}{\beta'}$ *from* P *and closed substitution* σ *such that* $\sigma(K') = K \wedge \sigma(\beta') = \beta$,
- *or* β *is negative and for every set* N *of closed negative literals such that* $\frac{N}{\gamma}$ *is irredundantly provable from* P *for* γ *a closed literal denying* β, *a literal in* K *denies one in* N.

A well-supported proof of α *from* P *(if it exists) is denoted by* $P \vdash_{ws} \alpha$.

In order to unequivocally define an LTS, a TSS has to be complete, as defined below.

Definition 8 (Complete TSSs). *A standard TSS is complete if and only if for any closed literal* $t \stackrel{a}{\not\rightarrow}$, *either* $P \vdash_{ws} t \stackrel{a}{\rightarrow} t'$ *for some closed term* t', *or* $P \vdash_{ws} t \stackrel{a}{\not\rightarrow}$.

It is often possible to establish completeness by using a syntactic measure on rules, called stratification [10]. All practical examples of TSSs are standard and complete and hence, almost all SOS meta-theorems are formed around complete TSSs. In this paper, we also follow this tradition and formulate our results for complete TSSs.

2.3 Rule Formats

The goal of a rule format is to establish a semantic property via syntactic constraints on rules. One of the most important semantic properties addressed by rule formats is compositionality or (pre-)congruence, defined below.

Definition 9 (Pre-congruence). *Let* P *be a TSS over signature* Σ. *A preorder* $\sqsubseteq \subseteq T(\Sigma) \times T(\Sigma)$ *on closed terms is a* pre-congruence *if and only if for all operators* $f \in \Sigma$ *and closed terms* $t_1, t'_1, \cdots, t_{ar(f)}, t'_{ar(f)} \in T(\Sigma)$, *we have that* $t_i \sqsubseteq t'_i$ *(for* $i \in [1, ar(f)]$*) implies* $f(t_1, \cdots, t_{ar(f)}) \sqsubseteq f(t'_1, \cdots, t'_{ar(f)})$.

A rule format that establishes pre-congruence for simulation (and congruence for bisimulation) is the following ntyft/ntyxt format [14].

Definition 10 (ntyft/ntyxt format). *An* ntytt *rule is a transition rule in which the right-hand sides of positive premises are variables that are all distinct and do not occur in the source of the conclusion. An ntytt rule is an* ntyxt *rule if the source of its conclusion is a variable and an* ntyft *rule if the source of its conclusion contains exactly one function symbol applied to distinct variables. An ntytt rule (resp. an ntyft rule) is an* nxytt *rule (resp. an* nxyft *rule) if the left-hand sides of its premises are variables. A TSS is in the* ntyft/ntyxt format *if it contains only ntyft and ntyxt rules.*

The ready simulation format, defined below, guarantees pre-congruence for ready simulation. Moreover, it is the basis of the modal decomposition technique presented in [9, 13] and hence, also serves as the basis of our approach.

Definition 11 (Ready simulation format). *A transition rule has no* lookahead *if the variables occurring in the right-hand sides of its positive premises do not occur in the left-hand sides of its premises. A TSS is in the* ready simulation format *if it is in the ntyft/ntyxt format and its transition rules have no lookahead.*

SOS rules are meant to define a flow of variable valuations from the source of the conclusion to the premises and eventually to the target of the conclusion. However, some rules may feature free variables whose valuations do not depend on the source of the conclusion. Rules without free variables and lookahead are called decent [9].

Definition 12 (Decent rule). *A variable occurring in a transition rule is* free *iff it does not occur in the source of the conclusion nor in the right-hand sides of the positive premises of the rule. A transition rule is* decent *if it has no lookahead and does not contain free variables.*

Rules with free variables can always be replaced with infinitely many decent rules, by replacing the free variables with all their possible closed valuations. The following lemma captures this intuition. According to the following lemma, focusing on decent rules in the proofs does not impose any extra theoretical constraint.

Lemma 2 ([9]). *Let P be a standard TSS in the ready simulation format. Then there is a TSS P^+ in the decent ntyft format such that any closed literal α is provable from P^+ if and only if $P \vdash_{ws} \alpha$.*

Definition 13. *A P-*ruloid *is a decent nxytt rule that is irredundantly provable from P^+. Lastly, the set of all P-ruloids of a given TSS P is denoted by $\bar{P}$.*

For the results to come, we need the following lemma. Intuitively, it states that for any TSS P in the ready simulation format, there is a well-supported proof of a positive closed literal α if and only if there is an irredundant proof of a P-ruloid such that the closed literal α is a closed substitution instance of the ruloid.

Lemma 3 ([9]). *Let P be a TSS in the ready simulation format. For any term $t \in \mathbb{T}(\Sigma)$, closed term t', and a closed substitution σ, we have $P \vdash_{ws} \sigma(t) \xrightarrow{a} t'$ iff there are a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' such that $P \vdash_{ws} \sigma'(\alpha)$ (for every $\alpha \in H$), $\sigma'(t) = \sigma(t)$, and $\sigma'(u) = t'$.*

3 Deriving a Pre-congruence Format

The basic machinery developed in [9] to derive a pre-congruence format works in two steps. First, a modal formula $\varphi \in \Phi$ for an open term t is decomposed into a choice of modal formulas $\psi(x)$ for variables x such that $\sigma(t)$ satisfies φ if and only if for one of those ψ's and all the variables x in t, $\sigma(x)$ satisfies $\psi(x)$ (Theorem 2). This is achieved by considering the provable transition rules for term t (given that such rules are in a given rule format.) Secondly a pre-congruence format for a pre-order is devised such that if a modal formula belongs to characterizing logic of the pre-order, then the resulting decomposed modal formulas also belong to the same characterizing logic (Theorem 3).

3.1 Modal Decomposition

Definition 14. *Let P be a standard TSS over Σ in the ready simulation format. The decomposition function $\cdot^{-1} : \mathbb{T}(\Sigma) \to (\Phi \to 2^{V \to \Phi})$ for a term is defined in the following way:*

1. *$\psi \in t^{-1}(\langle a \rangle \varphi)$ iff*

$$\psi(x) = \bigvee_{\frac{H}{t \xrightarrow{a} u} \in \bar{P}} \bigvee_{\chi \in u^{-1}(\varphi)} \Big(\chi(x) \wedge \bigwedge_{(x \xrightarrow{c} \not\rightarrow) \in H} [c]\bot \wedge \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \chi(y) \Big),$$

   whenever $x \in \mathbf{var}(t)$. For $x \notin \mathbf{var}(t)$, we let $\psi(x) = \top$.
2. *$\psi \in t^{-1}([a]\varphi)$ iff $\psi(x)$ (for $x \in \mathbf{var}(t)$) is defined to be*

$$\bigwedge_{\frac{H}{t \xrightarrow{a} u} \in \bar{P}} \Big[\Big(\bigwedge_{(x \xrightarrow{c} \not\rightarrow) \in H} [c]\bot \wedge \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \top \Big) \Rightarrow \Big(\bigvee_{\chi \in u^{-1}(\varphi)} \chi(x) \wedge \bigwedge_{(x \xrightarrow{b} y) \in H} [b] \bigvee_{\chi \in u^{-1}(\varphi)} \chi(y) \Big) \Big].$$

   As in the previous case, we let $\psi(x) = \top$ for $x \notin \mathbf{var}(t)$.
3. *$\psi \in t^{-1}(\bigwedge_{i \in I} \varphi_i)$ iff $\psi(x) = \bigwedge_{i \in I} \psi_i(x)$, where $\psi_i \in t^{-1}(\varphi_i)$ for $i \in I$.*
4. *$\psi \in t^{-1}(\bigvee_{i \in I} \varphi_i)$ iff $\psi(x) = \bigvee_{i \in I} \psi_i(x)$, where $\psi_i \in t^{-1}(\varphi_i)$ for $i \in I$.*

Note that item **2.** has not been treated in the past decomposition approaches [9, 13]. It concerns the semantic clause of the box modality $[a]\varphi$, i.e., for any closed terms t, t', if there is a transition $t \xrightarrow{a} t'$, then t' must satisfy φ.

Theorem 2. *Let P be a complete TSS in the ready simulation format over the signature Σ. Then, for any term $t \in \mathbb{T}(\Sigma)$, closed substitution σ, and a formula $\varphi \in \Phi$, we have $\sigma(t) \models \varphi \iff \exists_{\psi \in t^{-1}(\varphi)} \forall_{x \in \mathbf{var}(t)}\ \sigma(x) \models \psi(x)$.*

Proof. By structural induction on φ. In the remainder, we only consider the case when $\varphi = [a]\varphi'$. The proof of the remaining cases is the same as the proof given in [13, Theorem 2].

($\Leftarrow$) Let $\sigma(t) \xrightarrow{a} t'$ for some closed term t'. We need to show that $t' \models \varphi'$. We begin by using Lemma 3 to find a P-ruloid of the form:

$$\frac{\{x \xrightarrow{b_i} y_i \mid i \in I_x \wedge x \in \mathbf{var}(t)\} \cup \{x \xrightarrow{c_j} \mid j \in J_x \wedge x \in \mathbf{var}(t)\}}{t \xrightarrow{a} u} \tag{1}$$

and a closed substitution σ' such that $\sigma(t) = \sigma'(t)$, $P \vdash_{\mathrm{ws}} \sigma'(H)$, and $\sigma'(u) = t'$. Since $\exists_{\psi \in t^{-1}(\varphi)} \forall_{x \in \mathbf{var}(t)}\ \sigma(x) \models \psi(x)$, by Definition 14, we have (for every $x \in \mathbf{var}(t)$):

$$\sigma(x) \models \Big(\bigwedge_{j \in J_x} [c_j]\bot \wedge \bigwedge_{i \in I_x} \langle b_i \rangle \top \Big) \Rightarrow \Big(\bigvee_{\chi \in u^{-1}(\varphi')} \chi(x) \ \wedge \ \bigwedge_{i \in I_x} [b_i] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y) \Big). \tag{2}$$

We claim that $\forall_{z \in \mathbf{var}(u)}\ \sigma'(z) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(z)$. Let $z \in \mathbf{var}(u)$. We distinguish the following cases depending on the position of z in the decent P-ruloid:

- Let $z = x$ for some $x \in \mathbf{var}(t)$. Using $\sigma(x) = \sigma'(x)$ and $P \vdash_{\mathrm{ws}} \sigma'(H)$ in (2) we get $\sigma'(x) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(x)$.
- Let $z = y_i$ for some $i \in I_x$ and $x \in \mathbf{var}(t)$. Then, using $\sigma(x) = \sigma'(x)$ and $P \vdash_{\mathrm{ws}} \sigma'(H)$ in (2) we have $\sigma'(x) \models [b_i] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y_i)$ and $P \vdash_{\mathrm{ws}} \sigma'(x) \xrightarrow{b_i} \sigma'(y_i)$. Therefore, from the semantics of box modality we obtain $\sigma'(y_i) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y_i)$.

This proves the claim. Fix $\bar{\chi}(z) = \bigvee_{\chi \in u^{-1}(\varphi')} \chi(z)$ for every $z \in \mathbf{var}(u)$. Since Definition 14 is closed under arbitrary disjunctions, we know that $\bar{\chi} \in u^{-1}(\varphi')$. Moreover, we have $\sigma'(z) \models \bar{\chi}(z)$ (for every $z \in \mathbf{var}(u)$). Thus, by the induction hypothesis we obtain $\sigma'(u) \models \varphi'$.

($\Rightarrow$) Let $\sigma(t) \models [a]\varphi'$. Suppose there are no P-ruloids of the form $\frac{H}{t \xrightarrow{a} u}$. Then, by Definition 14 we have $\psi(x) = \bigwedge_{\emptyset} = \top$ for every $x \in \mathbf{var}(t)$. Since every closed term satisfies $\top$, we have $\sigma(x) \models \psi(x)$ for every $x \in \mathbf{var}(t)$ as required.

Now suppose there is a P-ruloid of the form given in (1). It suffices to show that the condition in (2) holds. Assume that $\sigma(x) \models \bigwedge_{j \in J_x} [c_j]\bot \wedge \bigwedge_{i \in I_x} \langle b_i \rangle \top$. Then, the completeness of P together with the semantics of box modality guarantee that $P \vdash_{\mathrm{ws}} \sigma(x) \xrightarrow{c_j}$ (for every $j \in J_x$). Furthermore, from the semantics of diamond modality, for every $i \in I_x$, we find some closed term t_i such that $P \vdash_{\mathrm{ws}} \sigma(x) \xrightarrow{b_i} t_i$. Thus, we can define a closed substitution σ' such that $\sigma(x) = \sigma'(x)$ (for $x \in \mathbf{var}(t)$), $\sigma'(y_i) = t_i$ (for $i \in I_x$). Note that σ' is well-defined because the P-ruloids have no lookahead and all y_i's are distinct

(i.e., $\forall_{i,i' \in I_x}\ i \neq i' \Rightarrow y_i \neq y_{i'}$). By Lemma 3, we obtain $\sigma(t) \xrightarrow{a} \sigma'(u)$. Thus, $\sigma'(u) \models \varphi'$ because $\sigma(t) \models [a]\varphi'$. From the induction hypothesis we obtain

$$\exists_{\chi \in u^{-1}(\varphi')} \forall_{z \in \mathbf{var}(u)}\ \sigma'(z) \models \chi(z). \tag{3}$$

From (3) we have, for every $x \in \mathbf{var}(t)$, $\sigma(x) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(x)$. Thus, it suffices to show that, for every $x \in \mathbf{var}(t)$, we have $\sigma(x) \models \bigwedge_{i \in I_x} [b_i] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y_i)$.

Let $\sigma(x) \xrightarrow{b_i} t''$ for some $i \in I_x$. Then, we define a closed substitution σ'' such that $\sigma(t) = \sigma''(t)$, $\sigma''(y_i) = t''$, and $\sigma''(y_{i'}) = \sigma'(y_{i'})$ (for $i' \in I_x$ such that $i \neq i'$). By repeating the same arguments (from above) to derive $P \vdash_{\mathrm{ws}} \sigma(t) \xrightarrow{a} \sigma'(u)$, we can find $P \vdash_{\mathrm{ws}} \sigma(t) \xrightarrow{a} \sigma''(u)$. Thus, $\sigma''(u) \models \varphi'$ because $\sigma(t) \models [a]\varphi'$. We can again instantiate the induction hypothesis to find a $\chi'' \in u^{-1}(\varphi')$ such that $\forall_{z \in \mathbf{var}(u)}\ \sigma''(z) \models \chi''(z)$. Therefore, $\sigma''(y_i) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y_i)$ and we can conclude that $\sigma(x) \models [b_i] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y_i)$.

We have shown for every P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and for every $x \in \mathbf{var}(t)$, if $\sigma(x) \models \bigwedge_{(x \xrightarrow{c} \not\rightarrow) \in H} [c]\bot$ and $\sigma(x) \models \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \top$ then $\sigma(x) \models \bigvee_{\chi \in u^{-1}(\varphi')} \chi(x)$ and $\sigma(x) \models \bigwedge_{x \xrightarrow{b} y \in H} [b] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y)$. Therefore, the formula $\psi(x)$ as defined in Definition 14(2) is satisfied by $\sigma(x)$. □

3.2 *XY*-simulation Format

Definition 15. *Given a set H of premises, we write H^+ and H^- to denote the set of all positive and negative literals in H, respectively. A rule $\frac{H}{t \xrightarrow{a} u}$ is in the* XY-simulation format *iff it is in the ready simulation format and the following conditions hold:*

1. *If $a \in X$ then*
   (a) $\forall_\alpha\ (\alpha \in H^+ \Rightarrow action(\alpha) \in X)$
   (b) $\forall_\alpha\ (\alpha \in H^- \Rightarrow action(\alpha) \in Y)$
2. *If $a \in Y$ then*
   (a) $\forall_\alpha\ (\alpha \in H^+ \Rightarrow action(\alpha) \in Y)$
   (b) $\forall_\alpha\ (\alpha \in H^- \Rightarrow action(\alpha) \in X)$

A TSS is in the XY-simulation format *iff all its rules are in the XY-simulation format.*

Lemma 4. *If a TSS is in the XY-simulation format, then all its P-ruloids are.*

Due to space limitations, we do not present the complete poof of Lemma 4. It goes by an induction on the depth of the irredundant proof for the P-ruloid at hand.[1]

Theorem 3. *Let P be a standard TSS in the XY-simulation format and Σ be its signature. If $t \in \mathbb{T}(\Sigma)$, $\varphi \in \Phi_{X,Y}$, and $\psi \in t^{-1}(\varphi)$ then $\forall_{x \in \mathbf{var}(t)}\ \psi(x) \in \Phi_{X,Y}$.*

[1] In this version of the paper, the proof is provided in the appendix.

Proof. We prove this theorem by structural induction on φ and consider the cases when $\varphi = \langle a \rangle \varphi'$ and $\varphi = [a]\varphi'$. In the following, due to Lemma 4, we use the fact that every derived P-ruloid is in the XY-simulation format, whenever P is in the XY-simulation format.
(1) Let $\varphi = \langle a \rangle \varphi'$ for some $a \in X$. By Definition 14, we have

$$\psi(x) = \Big(\chi(x) \wedge \bigwedge_{(x \overset{c}{\not\to}) \in H} [c]\bot \wedge \bigwedge_{(x \overset{b}{\to} y) \in H} \langle b \rangle \chi(y) \Big) \ ,$$

for some P-ruloid $\frac{H}{t \overset{a}{\to} u}$ and a decomposition function $\chi \in u^{-1}(\varphi')$. Hence, by the induction hypothesis $\chi(z) \in \Phi_{X,Y}$ (for $z \in \mathbf{var}(u)$). It suffices to show that $\forall_{(x \overset{c}{\not\to}) \in H} \ c \in Y$ and $\forall_{x \overset{b}{\to} y \in H} \ b \in X$.

- Let $(x \overset{c}{\not\to}) \in H$. Then, Definition 15(1b) ensures that $c \in Y$.
- Let $x \overset{b}{\to} y \in H$. Then, Definition 15(1a) ensures that $b \in X$.

(2) Let $\varphi = [a]\varphi'$ for some $a \in Y$. By Definition 14, we have (for $x \in \mathbf{var}(t)$):

$$\psi(x) = \bigwedge_{\frac{H}{t \overset{a}{\to} u} \in \bar{P}} \Big(\bigvee_{(x \overset{c}{\not\to}) \in H} \langle c \rangle \top \ \vee \bigvee_{(x \overset{b}{\to} y) \in H} [b]\bot \vee \Big(\bigvee_{\chi \in u^{-1}(\varphi')} \chi(x) \wedge \bigwedge_{(x \overset{b}{\to} y) \in H} [b] \bigvee_{\chi \in u^{-1}(\varphi')} \chi(y) \Big) \Big).$$

By the induction hypothesis we have, for every $\chi \in u^{-1}(\varphi'), z \in \mathbf{var}(u)$, that $\chi(z)$ is a formula in $\Phi_{X,Y}$; therefore $\bigvee_{\chi \in u^{-1}(\varphi')} \chi(z)$ is a formula in $\Phi_{X,Y}$. Thus, it suffices to show that $\forall_{x \overset{b}{\to} y \in H} \ b \in X \Rightarrow b \in Y$ and $\forall_{(x \overset{c}{\not\to}) \in H} \ c \in Y \Rightarrow c \in X$, which follow directly from conditions (2a) and (2b) of Definition 15, respectively. □

Corollary 1 (Main Result). *Let P be a complete TSS in the XY-simulation format over the signature Σ. Then, for any term $t \in \mathbb{T}(\Sigma)$ and closed substitutions σ, σ' we have: $\forall_{x \in \mathbf{var}(t)} \ \sigma(x) \preceq_{X,Y} \sigma'(x) \implies \sigma(t) \preceq_{X,Y} \sigma'(t)$.*

Proof. It suffices to show that if $\sigma(t) \models \varphi$ then $\sigma'(t) \models \varphi$, for all $\varphi \in \Phi_{X,Y}$.

$$\begin{aligned}
\sigma(t) \models \varphi &\implies \exists_{\psi \in t^{-1}(\varphi) \cap \Phi_{X,Y}} \forall_{x \in \mathbf{var}(t)} \ \sigma(x) \models \psi(x) && \text{(Theorem 2 and 3)} \\
&\implies \exists_{\psi \in t^{-1}(\varphi) \cap \Phi_{X,Y}} \forall_{x \in \mathbf{var}(t)} \ \sigma'(x) \models \psi(x) && (\because \forall_{x \in \mathbf{var}(t)} \ \sigma(x) \preceq_{X,Y} \sigma'(x)) \\
&\implies \sigma'(t) \models \varphi && \text{(Theorem 2).}
\end{aligned}$$

4 Applications

In this section, we review the different incarnations of XY-simulation relation present in the literature and assert their pre-congruence property with respect to some well-known operators from the field of process algebra. To start with, through the following proposition, we establish a link between XY-similarity and some other notions of behavioral pre-order and equivalence.

Proposition 1. *Let $(\mathbb{P}, \mathcal{A}, \rightarrow)$ be an arbitrary LTS. Then, the following statements hold:*

1. *Relation $\preceq_{\mathcal{A},\mathcal{A}}$ is the bisimilarity relation in the sense of [20].*
2. *Relation $\preceq_{\mathcal{A},\emptyset}$ is the similarity relation in the sense of [18].*
3. *If $X \subseteq \mathcal{A}$, then the relation $\preceq_{\mathcal{A},X}$ is the partial bisimilarity relation in the sense of [6].*
4. *If the set of actions are partitioned into two sets of may actions $\mathcal{A}_\Diamond$ and must actions $\mathcal{A}_\Box$, then the relation $\preceq_{\mathcal{A}_\Diamond,\mathcal{A}_\Box}$ is the modal refinement relation in the sense of [16].*
5. *If the set of actions are partitioned into two sets of input actions $\mathcal{I}$ and output actions $\mathcal{O}$, then the relation $\preceq_{\mathcal{O},\mathcal{I}}$ is the alternating similarity relation in the sense of [4].*

In the following subsection, we show how our rule format can be applied to obtain compositionality results for various process calculi.

4.1 Partial Bisimulation

In [6], Baeten et al. used the partial bisimulation pre-order to define controllability of nondeterministic processes. (Controllability is a central notion in the supervisory control theory.) To this end, they defined a basic sequential process algebra $\mathrm{BSP}_|(\mathcal{A}_\downarrow, B)$ (for some fixed subset $B \subseteq \mathcal{A}$ and $\mathcal{A}_\downarrow = \mathcal{A} \uplus \{\downarrow\}$[2]) and provided a ground-complete axiomatization of partial bisimulation pre-order. The signature of process terms $\mathbf{\Sigma}$ in $\mathrm{BSP}_|(\mathcal{A}_\downarrow, B)$ is given below:

$$\mathbf{\Sigma} = \{\ (\mathbf{0}, 0)\ ,\ (\mathbf{1}, 0)\ ,\ (a., 1)_{a\in\mathcal{A}}\ ,\ (+, 2)\ ,\ (|, 2)\ \}\,.$$

Constant $\mathbf{0}$, called *inaction*, denotes that no actions can be performed and can only deadlock, whereas constant $\mathbf{1}$ denotes successful termination. The family of unary operators $a.\_$ (for $a \in \mathcal{A}$), called *action prefix* operator, expresses that a process can initially perform a and then the argument process takes over. Binary operator $\_ + \_$, known as the *alternative composition* operator, specifies the choice between two process terms. Lastly, the *synchronization parallel composition* is denoted by $\_|\_$ and specifies that the two arguments synchronize on common actions. The formal semantics for each operator in $\mathbf{\Sigma}$ is given in Table 1 by means of a standard TSS that is in the ready simulation format.

By a quick inspection of the labels, we note that all rules in Table 1 are in the $\mathcal{A}_\downarrow B$-simulation format, the $\mathcal{A}_\downarrow\emptyset$-simulation format, and the $\mathcal{A}_\downarrow\mathcal{A}_\downarrow$-simulation format. Therefore, we obtain the following (pre-)congruence results for free.

Corollary 2. *Partial bisimilarity pre-order $\preceq_{\mathcal{A}_\downarrow,B} \subseteq T(\mathbf{\Sigma}) \times T(\mathbf{\Sigma})$ is a pre-congruence relation for all closed terms in process algebra $BSP_|(\mathcal{A}_\downarrow, B)$. Moreover, similarity pre-order $\preceq_{\mathcal{A}_\downarrow,\emptyset}$ and bisimilarity equivalence $\preceq_{\mathcal{A}_\downarrow,\mathcal{A}_\downarrow}$ are also pre-congruence and congruence relations, respectively, for all constructs of process algebra $BSP_|(\mathcal{A}_\downarrow, B)$.*

[2] We employ $\downarrow$ (by a coding proposed by Baeten and Verhoef in [7]) as a special action label modeling successful termination.

Table 1. Operational rules of $\mathrm{BSP}_{|}(\mathcal{A}_{\downarrow}, B)$, where $a \in \mathcal{A}, a_{\downarrow} \in \mathcal{A} \cup \{\downarrow\}$.

$$\mathbf{1} \xrightarrow{\downarrow} \mathbf{1} \quad (1) \qquad a.x \xrightarrow{a} x \quad (2) \qquad \frac{x \xrightarrow{a_{\downarrow}} x'}{x + y \xrightarrow{a_{\downarrow}} x'} \quad (3)$$

$$\frac{y \xrightarrow{a_{\downarrow}} y'}{x + y \xrightarrow{a_{\downarrow}} y'} \quad (4) \qquad \frac{x \xrightarrow{a_{\downarrow}} x' \quad y \xrightarrow{a_{\downarrow}} y'}{x|y \xrightarrow{a_{\downarrow}} x'|y'} \quad (5)$$

4.2 Modal Refinement

Next, we consider the framework of modal specifications [15, 16]. Let Act be the set of action labels ranged over by $\mathbf{a}, \mathbf{b}, \cdots$. Construct the set of *may* and *must* actions as: $\mathcal{A}_{\Diamond} = Act \times \{\Diamond\}$ and $\mathcal{A}_{\Box} = Act \times \{\Box\}$. We write $\mathbf{a}_{\Diamond}$ and $\mathbf{a}_{\Box}$ to denote the elements $(\mathbf{a}, \Diamond) \in \mathcal{A}_{\Diamond}$ and $(\mathbf{a}, \Box) \in \mathcal{A}_{\Box}$, respectively. Let $\mathcal{A} = \mathcal{A}_{\Diamond} \cup \mathcal{A}_{\Box}$ and consider the following signature:

$$\boldsymbol{\Sigma}_m = \{\ (\mathbf{0}, 0)\ ,\ (a., 1)_{a \in \mathcal{A}}\ ,\ (+, 2)\ ,\ (|, 2)\ ,\ (\vee, 2)\ ,\ (\wedge, 2)\ \}\,.$$

The formal semantics of the operators in $\boldsymbol{\Sigma} \cap \boldsymbol{\Sigma}_m$ remains the same in this new setting, whereas the semantics of conjunction and disjunction is given by the rules in Table 2.

Table 2. Operational rules for $\vee$ and $\wedge$, taken from [15]

$$\frac{x \xrightarrow{\mathbf{a}_{\Diamond}} x'}{x \vee y \xrightarrow{\mathbf{a}_{\Diamond}} x'} \quad (6) \qquad \frac{y \xrightarrow{\mathbf{a}_{\Diamond}} y'}{x \vee y \xrightarrow{\mathbf{a}_{\Diamond}} y'} \quad (7) \qquad \frac{x \xrightarrow{\mathbf{a}_{\Box}} x' \quad y \xrightarrow{\mathbf{a}_{\Box}} y'}{x \vee y \xrightarrow{\mathbf{a}_{\Box}} x' \vee y'} \quad (8)$$

$$\frac{x \xrightarrow{\mathbf{a}_{\Box}} x'}{x \wedge y \xrightarrow{\mathbf{a}_{\Box}} x'} \quad (9) \qquad \frac{y \xrightarrow{\mathbf{a}_{\Box}} y'}{x \wedge y \xrightarrow{\mathbf{a}_{\Box}} y'} \quad (10) \qquad \frac{x \xrightarrow{\mathbf{a}_{\Diamond}} x' \quad y \xrightarrow{\mathbf{a}_{\Diamond}} y'}{x \wedge y \xrightarrow{\mathbf{a}_{\Diamond}} x' \wedge y'} \quad (11)$$

Note that the process terms induced by our operational rules are not admissible (consistent) in the sense of [16], i.e., the set of must transitions are not necessary included in the set of may transitions. In essence, the transition system induced by our algebra corresponds to the mixed transition system, where the consistency assumption is dropped.

By inspection we note that all the rules in Table 1 and Table 2 are in $\mathcal{A}_{\Diamond}\mathcal{A}_{\Box}$-simulation format. Therefore, we obtain the following pre-congruence result for free.

Corollary 3. *The modal refinement pre-order $\preceq_{\mathcal{A}_{\Diamond},\mathcal{A}_{\Box}} \subseteq T(\boldsymbol{\Sigma}_m) \times T(\boldsymbol{\Sigma}_m)$ is a pre-congruence relation. Moreover, the $\mathcal{A}_{\Diamond}\mathcal{A}_{\Box}$-simulation format subsumes the static constructor format given by Larsen and Thomsen [16, Section 4].*

Next consider the following modified operational rules of conjunction $\wedge'$ taken from [17]. Note that, in [17], the conjunction is defined between any two arbitrary interface automata [12] and we interpret the input actions as must actions and the output actions as may actions.

$$\frac{x \xrightarrow{\mathbf{a}_\Box} x' \quad y \xrightarrow{\mathbf{a}_\Box}}{x \wedge' y \xrightarrow{\mathbf{a}_\Box} x'} \quad (9') \qquad \frac{y \xrightarrow{\mathbf{a}_\Box} y' \quad x \xrightarrow{\mathbf{a}_\Box}}{x \wedge' y \xrightarrow{\mathbf{a}_\Box} y'} \quad (10')$$

$$\frac{x \xrightarrow{\mathbf{a}_\Box} x' \quad y \xrightarrow{\mathbf{a}_\Box} y'}{x \wedge' y \xrightarrow{\mathbf{a}_\Box} x' \wedge' y'} \quad (11') \qquad \frac{x \xrightarrow{\mathbf{a}_\Diamond} x' \quad y \xrightarrow{\mathbf{a}_\Diamond} y'}{x \wedge' y \xrightarrow{\mathbf{a}_\Diamond} x' \wedge' y'} \quad (11'')$$

Clearly, rules $(9')$ and $(10')$ are not in the $\mathcal{A}_\Diamond\mathcal{A}_\Box$-simulation format because they violate condition (2b) of Definition 15. Next, by a counterexample, we show that the modal refinement pre-order is not a pre-congruence for the modified conjunction operator $\wedge'$.

Example 1. Consider the following process terms: $t = \mathbf{a}_\Box.\mathbf{b}_\Box.0$, $t' = \mathbf{a}_\Box.\mathbf{c}_\Box.0$, and $\bar{t} = t + t'$. Clearly, $\bar{t} \preceq_{\mathcal{A}_\Diamond,\mathcal{A}_\Box} t$ and $\bar{t} \preceq_{\mathcal{A}_\Diamond,\mathcal{A}_\Box} t'$. However, $\bar{t} \wedge' \bar{t} \not\preceq_{\mathcal{A}_\Diamond,\mathcal{A}_\Box} t \wedge' t'$.

5 Adequacy of XY-simulation Format

In this section, with the help of the following counterexamples, we motivate why the conditions of XY-simulation format are essential for the pre-congruence result. In particular, we show how dropping each of the conditions is sufficient for breaking pre-congruence.

Example 2. Consider the synchronous parallel composition parameterized with a partial function $\gamma : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ (called as *communication* function [5]) such that rule 5 is substituted by the following rules:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad \gamma(a,b) \text{ is defined}}{x|_\gamma y \xrightarrow{\gamma(a,b)} x'|_\gamma y'} \quad (5') \qquad \frac{x \xrightarrow{\downarrow} x' \quad y \xrightarrow{\downarrow} y'}{x|_\gamma y \xrightarrow{\downarrow} x'|_\gamma y'} \quad (5'').$$

Let $\mathcal{A} = \{a, b\}$ and the communication function γ be defined as: $\gamma(b, b) = a$ and undefined otherwise. Clearly, the inequation $b.\mathbf{0} \preceq_{\{a\},\{b\}} a.\mathbf{0}$ holds; however, $b.\mathbf{0}|_\gamma b.\mathbf{0} \not\preceq_{\{a\},\{b\}} a.\mathbf{0}|_\gamma a.\mathbf{0}$. We note that rule 5 of $|_\gamma$ violates Definition 15(1a). Similarly, by defining a communication function γ' as $\gamma(a, a) = b$ and undefined otherwise, we can see that $b.\mathbf{0}|_{\gamma'} b.\mathbf{0} \not\preceq_{\{a\},\{b\}} a.\mathbf{0}|_{\gamma'} a.\mathbf{0}$. Furthermore, we now note that rule 5 of $|_{\gamma'}$ violates Definition 15(2a).

Example 3. This example concerns negative premises. Consider the unary operator θ (called the *priority* operator) from TCP [5], which also comes with a partial ordering $<$ on the set of actions $\mathcal{A}$. Intuitively, the priority operator can execute an a-transition if the operand can execute an a-transition and no action with priority over a can be executed.

$$\frac{x \xrightarrow{a} x' \quad x \overset{b}{\nrightarrow} \quad \text{for all } b \text{ with } a < b}{\theta(x) \xrightarrow{a} \theta(x')} \quad (12)$$

Clearly, the above rule is in the ready simulation format. Let $\mathcal{A} = \{a, b\}$ with $a < b$ and consider the process terms $a.\mathbf{0}, a.\mathbf{0} + b.\mathbf{0}$. It holds that $a.\mathbf{0} \preceq_{\mathcal{A},\emptyset} a.\mathbf{0} + b.\mathbf{0}$; however, $\theta(a.\mathbf{0}) \not\preceq_{\mathcal{A},\emptyset} \theta(a.\mathbf{0} + b.\mathbf{0})$. We note that rule 12 of θ violates Definition 15(1b). Furthermore, since $\preceq_{\emptyset,X} = \preceq^{-1}_{X,\emptyset}$, the above counterexample also highlights the violation of Definition 15(2b).

6 Conclusions

In this paper, we proposed a pre-congruence rule format for XY-simulation. The rule format guarantees that once the SOS rules of a given language satisfy certain syntactic conditions, then XY-simulation is pre-congruence for the constructs of the language. We showed that the format is applicable to obtain compositionality results for different behavioral pre-orders and for different process calculi. We also showed that dropping each of the syntactic conditions imposed by the rule format can jeopardize compositionality.

We intend to exploit the results of this paper in order to obtain a rule format for input-output conformance (ioco) testing [22], which is a behavioral pre-order widely used as a basis for model-based testing. This will generalize the earlier compositionality results reported in [8], which only address a particular synchronization operator and the hiding (abstraction) operator.

References

1. F. Aarts and F.W. Vaandrager. Learning I/O automata. In *Proc. of CONCUR 2010*, volume 6269 of *LNCS*, pages 71–85. Springer-Verlag, 2010.
2. L. Aceto, I. Fábregas, D. de Frutos Escrig, A. Ingólfsdóttir, and M. Palomino. Relating modal refinements, covariant-contravariant simulations and partial bisimulations. In *Proc. of FSEN 2011*, volume 7141 of LNCS, pages 268–283. Springer-Verlag, 2012.
3. L. Aceto, W.J. Fokkink, and C. Verhoef. Structural operational semantics. *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier, 2001.
4. R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *Proc. of CONCUR'98*, volume 1466 of LNCS, pages 163–178. Springer-Verlag, 1998.
5. J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press, 2009.
6. J.C.M. Baeten, D.A. van Beek, B. Luttik, J. Markovski, and J.E. Rooda. A process-theoretic approach to supervisory control theory. In *American Control Conference (ACC)*, pages 4496–4501, June 2011.
7. J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In Eike Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *LNCS*, pages 477–492. Springer-Verlag, 1993.

8. M. van der Bijl, A. Rensink, and J. Tretmans. Compositional testing with ioco. In *Formal Approaches to Software Testing*, volume 2931 of *LNCS*, pages 86–100. Springer-Verlag, 2004.
9. B. Bloom, W. Fokkink, and R.J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM ToCL*, 5(1):26–78, 2004.
10. R. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *J. ACM*, 43(5):863–914, September 1996.
11. G. Boudol and K.G. Larsen. Graphical versus logical specifications. *TCS*, 106(1):3 – 20, 1992.
12. L. de Alfaro and T.A. Henzinger. Interface automata. In *Proc. of ESEC/FSE-9*, pages 109–120. ACM, 2001.
13. W.J. Fokkink, R.J. van Glabbeek, and P. de Wind. Compositionality of hennessy-milner logic by structural operational semantics. *TCS*, 354(3):421 – 440, 2006.
14. J.F. Groote. Transition system specifications with negative premises. *TCS*, 118(2):263–299, 1993.
15. K.G. Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 232–246. Springer-Verlag, 1990.
16. K.G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 203–210, 1988.
17. G. Lüttgen and W. Vogler. Modal interface automata. In *Proc. of TCS'12*, pages 265–279. Springer-Verlag, 2012.
18. R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, IJCAI, pages 481–489. Morgan Kaufmann Publishers Inc., 1971.
19. M.R. Mousavi, M.A. Reniers, and J.F. Groote. SOS rule formats and meta-theory: 20 years after. *TCS*, 373:238 – 272, 2007.
20. D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on TCS*, pages 167–183. Springer-Verlag, 1981.
21. G.D. Plotkin. A structural approach to operational semantics. *JLAP*, 60:17–139, 2004.
22. J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *LNCS*, pages 1–38. Springer-Verlag, 2008.
23. R.J. van Glabbeek. The meaning of negative premises in transition system specifications II. *JLAP*, 60-61(0):229 – 258, 2004.

A Proof of Lemma 4

Proof. Assume that a TSS is in the XY-simulation format and consider a $P \in \bar{P}$; we prove by induction on the depth of the irredundant proof for P. The base case, where the irredundant proof has depth zero, can be split into two cases:

- Either P is of the form $\frac{}{t \xrightarrow{a} t'}$; then, the lemma follows immediately, since H is the empty set and hence, it satisfies the conditions of Definition 15 vacuously.
- Or P is of the form $\frac{x \xrightarrow{a} y}{x \xrightarrow{a} y}$; then, a and $action(\alpha)$ in Definition 15 coincide and since the only premise is a positive literal, the lemma holds.

For the induction step, consider the case where P has an irredundant proof of depth $n+1$, and assume that for all P' with shallower irredundant proofs, the lemma holds. Assume that P is of the form $\frac{\{x \xrightarrow{b_i} y_i \mid i \in I_x \wedge x \in \mathbf{var}(t)\} \cup \{x \xrightarrow{c_j} \mid j \in J_x \wedge x \in \mathbf{var}(t)\}}{t \xrightarrow{a} u}$. Without loss of generality, we assume that $a \in X$ and it remains to show that for each $i \in I_x$, $b_i \in X$ and for each $j \in J_x$, $c_j \in Y$.

Since the proof tree has a depth of at least 2, the root of proof tree is labelled $t \xrightarrow{a} u$ and the non-empty set of nodes above the root are labeled with formulae in a set H such that $\frac{H}{t \xrightarrow{a} u}$ is an instance of a deduction rule $\frac{H'}{t' \xrightarrow{a} u'}$ in the TSS with substitution σ applied to it, i.e., $\sigma(t') = t$, $\sigma(u') = u$, and $\sigma(H') = H$. Consider an arbitrary literal α among the premises of P; we distinguish the following two cases based on whether α is positive or negative:

- Positive: Consider an arbitrary $i \in I_x$ and $\alpha = x \xrightarrow{b}_i y_i$ among the premises of P; we distinguish the following cases based on the position of the node labeled α in the proof tree for P (note that because of the form of α, no node appears above the node(s) which is (are) labeled with α):
  - Either α appears as the label of a node just above the root (i.e., in a node of depth 2); in this case, $\alpha = \sigma(\alpha')$, for some $\alpha' \in H'$ which if of form $x' \xrightarrow{b_i} y'_i$ for some variables $x'_i \in \mathbf{var}(t')$ and y'_i. It follows from the fact that $\frac{H'}{t' \xrightarrow{a} u'}$ is in the TSS and that the TSS is in the XY-simulation format that $b_i \in X$, which was to be shown.
  - Or α only appears as the label of a node with depth 3 or more. Consider a premise $\beta \in H$ such that α appears as a label of a node above β. Note that β has to be a positive literal (since negative literals are necessarily among the hypotheses); moreover, since the deduction rule used to instantiate the first step of the proof is in the XY-simulation format, we have that $action(\beta) \in X$. The sub-tree rooted in β provides an irredundant proof for $\frac{H''}{\beta}$, where H'' is the set of all labels of the nodes in the sub-tree that do not have any other node above them. Hence, we have that $\alpha \in H''$ and by the induction hypothesis, given $action(\beta) \in X$, we have that $b_i \in X$, which was to be shown.

– Negative: Consider an arbitrary $j \in J_x$ and $\alpha = x \overset{G}{\not\Rightarrow}_j$ among the premises of P; similar to the previous case, we consider a node labeled α in the proof tree and distinguish two cases based on the position of the node; from an identical reasoning as to the above given two items it follows that $c_j \in Y$.