Logical characterisations and compositionality of input-output conformance simulation[⋆]

Luca Aceto[1], Ignacio Fábregas[1], Carlos Gregorio-Rodríguez[2], and Anna Ingólfsdóttir[1]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Iceland
[2] Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain

Abstract. Input-output conformance simulation (iocos) has been proposed by Gregorio-Rodríguez, Llana and Martínez-Torres as a simulation-based behavioural preorder underlying model-based testing. This relation is inspired by Tretman's classic ioco relation, but has better worst-case complexity than ioco and supports stepwise refinement. The goal of this paper is to develop the theory of iocos by studying logical characterisations of this relation and its compositionality. More specifically, this article presents characterisations of iocos in terms of modal logics and compares them with an existing logical characterisation for ioco proposed by Behoar and Mousavi. A precongruence rule format for iocos and a rule format ensuring that operations take quiescence properly into account are also given. Both rule formats are based on the GSOS format by Bloom, Istrail and Meyer.

1 Introduction

Model-based testing (MBT) is an increasingly popular technique for validation and verification of computing systems, and provides a compromise between formal verification approaches, such model checking, and manual testing. MBT uses a model to describe the aspects of system behaviour that are considered to be relevant at some suitable level of abstraction. This model is employed to generate test cases automatically, while guaranteeing that some coverage criterion is met. Such test cases are then executed on the actual system in order to check whether its behaviour complies with that described by the model.

A formal notion of compliance relation between models (specifications) and systems (implementations) provides a formal underpinning for MBT. The defacto standard compliance relation underlying MBT for labelled transition systems with input and output actions is the classic ioco relation proposed by Tretmans, for which a whole MBT framework and tools have been developed. (See, for instance, [16] and the references therein.)

⋆ Research partially supported by the Spanish projects TIN2015-65845-C3-1-R (MINECO/FEDER) and S2013/ICE-3006, the project 001-ABEL-CM-2013 within the NILS Science and Sustainability Programme and the project Nominal SOS (project nr. 141558-051) of the Icelandic Research Fund

An alternative conformance relation that can be used to underlie MBT is *input-output conformance simulation* ($\mathsf{ioco\underline{s}}$). This relation follows many of the ideas in the definition of ioco. However, $\mathsf{ioco\underline{s}}$ is a branching-time semantics based on simulation, whereas ioco is a trace-based semantics. $\mathsf{ioco\underline{s}}$ has been introduced, motivated and proved to be an adequate conformance relation for MBT in [8–10].

Since $\mathsf{ioco\underline{s}}$ has been proposed as an alternative, branching-time touchstone relation for MBT, it is natural to investigate its theory in order to understand its properties. The goal of this paper is to contribute to this endeavour by studying the discriminating power of $\mathsf{ioco\underline{s}}$ and its compositionality. More precisely, in Section 3, we provide modal characterisations of $\mathsf{ioco\underline{s}}$ in the style of Hennessy and Milner [11]. We offer two modal chacterisations of $\mathsf{ioco\underline{s}}$, which are based on the use of either a 'non-forcing diamond modality' (Theorem 1) or of a 'forcing box modality' (Theorem 2), and compare them with an existing logical characterisation for ioco proposed by Behoar and Mousavi in [3] (Section 4). We also show, by means of an example, that, contrary to what is claimed in [13, Theorem 2], ioco and $\mathsf{ioco\underline{s}}$ do *not* coincide even when implementations are input enabled (Section 4.1).

As argued in [2, 17] amongst other references, MBT can benefit from a compositional approach whose goal is to increase the efficiency of the testing activity. The above-mentioned references study compositionality of ioco with respect to a small collection of well-chosen operations. Here we take a general approach to the study of compositionality of $\mathsf{ioco\underline{s}}$, which is based on the theory of rule formats for structural operational semantics [1]. In Section 5, we present a congruence rule format for $\mathsf{ioco\underline{s}}$ based on the GSOS format proposed by Bloom, Istrail and Meyer [5] (Theorem 4). Since operations preserving $\mathsf{ioco\underline{s}}$ need to take quiescence properly into account, we also propose a rule format guaranteeing that operations preserve coherent quiescent behaviour (Theorem 5 in Section 5.1).

Section 6 concludes the paper and presents avenues for future research.

2 Preliminaries

The input-output conformance simulation preorder presented in [8–10, 13] (henceforth referred to as $\mathsf{ioco\underline{s}}$) is a semantic relation developed under the assumption that systems have two kinds of transitions: input actions, namely those that the systems are willing to admit or respond to, and output actions, which are those produced by the system and that can be seen as responses or results.

We use I to denote the alphabet of input actions, which are written with a question mark $(a?, b?, c? \ldots)$. We call O the alphabet of output actions, which are annotated with an exclamation mark $(a!, b!, \delta! \ldots)$. In many cases we want to name actions in a general sense, inputs and outputs indistinctly. We will consider the set $L = I \cup O$ and we will omit the exclamation or question marks when naming generic actions, $a, b, x, y, z \in L$.

A state with no output actions cannot proceed autonomously; such a state is called *quiescent*. Following Tretmans (see, for instance, [14, 16]), we directly introduce the event of quiescence as a special output action denoted by $\delta! \in O$ in the definition of our models.

Definition 1. *A* labelled transition system with inputs and outputs*, LTS for short, is a quadruple $(S, I, O, \rightarrow)$ such that*

- *S is a set of states, processes, or behaviours.*
- *I and O are disjoint sets of input and output actions, respectively. Output actions include the quiescence symbol $\delta! \in O$. We define $L = I \cup O$.*
- *$\rightarrow \subseteq S \times L \times S$ is the transition relation. As usual we write $p \xrightarrow{x} q$ instead of $(p, x, q) \in \rightarrow$ and $p \xrightarrow{x}$, for $x \in L$, if there exists some $q \in S$ such that $p \xrightarrow{x} q$. Analogously, we will write $p \not\xrightarrow{x}$, for $x \in L$, if there is no q such that $p \xrightarrow{x} q$.*
  In order to allow only for coherent quiescent systems, the set of transitions $\rightarrow$ should also satisfy the following requirement: $p \xrightarrow{\delta!} p'$ iff $p = p'$ and $p \not\xrightarrow{o!}$ for each $o! \in O \setminus \{\delta!\}$.

The extension of the transition relation to sequences of actions is defined as usual.

In general we use $p, q, p', q' \ldots$ for states or behaviours, but also i, i', s and s' when we want to emphasise the concrete role of a behaviour as an implementation or a specification, respectively. We consider implementations and specifications, or, more generally, behaviours under study, as states of the same LTS.

The formal definition of $\mathsf{ioco}\underline{\mathsf{s}}$ uses the following functions over states of an LTS:

$$\mathsf{outs}(p) = \{o! \mid o! \in O,\ p \xrightarrow{o!}\}, \text{ the set of initial outputs of a state } p.$$
$$\mathsf{ins}(p) = \{a? \mid a? \in I,\ p \xrightarrow{a?}\}, \text{ the set of initial inputs of a state } p.$$

Definition 2. *We say that a binary relation R over states in an LTS is an $\mathsf{ioco}\underline{\mathsf{s}}$-relation if, and only if, for each $(p, q) \in R$ the following conditions hold:*

1. *$\mathsf{ins}(q) \subseteq \mathsf{ins}(p)$.*
2. *For all $a? \in \mathsf{ins}(q)$ and $p' \in S$, if $p \xrightarrow{a?} p'$ then there exists some q' such that $q \xrightarrow{a?} q'$ with $(p', q') \in R$.*
3. *For all $o! \in O$ and $p' \in S$, if $p \xrightarrow{o!} p'$ then there exists some q' such that $q \xrightarrow{o!} q'$ with $(p', q') \in R$.*

We define the input-output conformance simulation *($\mathsf{ioco}\underline{\mathsf{s}}$) as the largest $\mathsf{ioco}\underline{\mathsf{s}}$-relation. We write $p \;\mathsf{ioco}\underline{\mathsf{s}}\; q$ instead of $(p, q) \in \mathsf{ioco}\underline{\mathsf{s}}$.*

Example 1. Consider the following processes:

$$\begin{tikzcd} i \arrow[loop left, "\delta!"] \arrow[loop right, "a?"] & & s \arrow[loop right, "\delta!"] \end{tikzcd}$$

It is easy to see that $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$. Indeed, $\mathsf{ins}(s) = \emptyset$ and therefore the specification s does not prevent the implementation i from offering the input transition $i \xrightarrow{a?} i$.

Throughout the paper we make extensive use of modal logics. A logic over processes is defined by a language to express the formulae and a satisfaction relation that defines when a process (that is, a state of an LTS) has the property described by some formula. A classic example and a reference for the rest of the paper is Hennessy-Milner Logic [11].

Definition 3. *Hennessy-Milner Logic over the set of actions L (abbreviated to HML) is the collection of formulae defined by the following BNF grammar:*

$$\phi ::= \mathbf{tt} \mid \mathbf{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\,a\,]\phi \mid \langle a \rangle \phi,$$

where $a \in L$. HML is interpreted over an LTS by defining a satisfaction relation $\models$ relating states to formulae. The semantics of the boolean constants $\mathbf{tt}$ and $\mathbf{ff}$ and of the boolean connectives $\wedge$ and $\vee$ is defined as usual. The satisfaction relation for the modalities $\langle a \rangle$ and $[\,a\,]$ is as follows:

- *$p \models \langle a \rangle \varphi$ iff there exists some p' such that $p \xrightarrow{a} p'$ and $p' \models \varphi$.*
- *$p \models [\,a\,]\varphi$ iff $p' \models \varphi$ for all p' such that $p \xrightarrow{a} p'$.*

Every subset of HML naturally induces a preorder on a given set of behaviours.

Definition 4. *Given a logic $\mathcal{L}$ included in HML and a set S of states in an LTS, we define $\leq_{\mathcal{L}}$ as the binary relation over S given by*

$$p \leq_{\mathcal{L}} q \quad \textit{iff} \quad \forall \phi \in \mathcal{L} \;\; (p \models \phi \Rightarrow q \models \phi).$$

Remark 1. In what follows, we will consider only *image-finite* LTSs, that is, LTSs where for each p and each $a \in I \cup O$ there are only finitely many p' such that $p \xrightarrow{a} p'$. Also, we will consider both I and O to be finite sets.

3 Logic for $\mathsf{ioco}\underline{\mathsf{s}}$

In this section we present a logic that characterises the $\mathsf{ioco}\underline{\mathsf{s}}$ relation. This logic is a subset of Hennessy-Milner Logic and is rather minimal, but is convenient to characterize clearly the discriminating power of the $\mathsf{ioco}\underline{\mathsf{s}}$ relation.

Definition 5. *The syntax of the logic for $\mathsf{ioco}\underline{\mathsf{s}}$, denoted by $\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$, is defined by the following BNF grammar:*

$$\phi ::= \mathbf{tt} \mid \mathbf{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\!| a? |\!\rangle \phi \mid \langle x! \rangle \phi,$$

where $a? \in I$ and $x! \in O$. The semantics of the constants $\mathbf{tt}$ and $\mathbf{ff}$, of the boolean connectives $\wedge$ and $\vee$, and of the modality $\langle x! \rangle$ is defined as usual. The satisfaction relation for the modality $\langle\!| a? |\!\rangle$ is given below:

- *$p \models \langle\!| a? |\!\rangle \phi$ iff $p \overset{a?}{\not\longrightarrow}$ or $p' \models \phi$ for some $p \xrightarrow{a?} p'$.*

The new modal operator $\langle\!| a? |\!\rangle$ can be read as a *non forcing* diamond modality: if the action specified in the modality is not possible in a given state then the formula is satisfied. This operator can be expressed with the classic modalities in HML; indeed, $\langle\!| a? |\!\rangle\phi$ is equivalent to $\langle a? \rangle\phi \vee [\,a?\,]\mathbf{ff}$. The need for this special modality arises because, in order for $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$ to hold, s need only match the input transitions of i that are labelled with input actions that s affords.

According to Definition 4, the logic $\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$ induces the preorder $\leq_{\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}}$. Next we prove that this logical preorder coincides with the input output conformance simulation preorder, $\mathsf{ioco}\underline{\mathsf{s}}$, over an arbitrary (image-finite) LTS.

Theorem 1. *For all states i, s in some LTS,*

$$i \;\mathsf{ioco}\underline{\mathsf{s}}\; s \quad \textit{iff} \quad i \leq_{\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}} s.$$

The proof of this result may be found in Appendix A.

The logic for $\mathsf{ioco}\underline{\mathsf{s}}$ we have presented in Definition 5 follows a standard approach to the logical characterisation of simulation semantics; see, for instance, [6, 18]. However, the $\mathsf{ioco}\underline{\mathsf{s}}$ relation originated in the model-based testing environment where the natural reading for a logical characterisation would be *'every property satisfied by the specification should also hold in the implementation'*. Next we define an alternative logic that better matches this specification/implementation view.

Definition 6. *The syntax of the logic $\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ is defined by the following BNF grammar:*

$$\phi ::= \mathbf{tt} \mid \mathbf{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\![a?]\!]\phi \mid [\,x!\,]\phi,$$

where $a? \in I$ and $x! \in O$. The semantics of the constants $\mathbf{tt}$ and $\mathbf{ff}$, of the boolean connectives $\wedge$ and $\vee$, and of the modality $[\,x!\,]$ is defined as usual. The satisfaction relation for the modalities $[\![a?]\!]$ is as follows:

– $p \models [\![a?]\!]\phi$ *iff* $p \xrightarrow{a?}$ *and* $p' \models \phi$, *for each* $p \xrightarrow{a?} p'$.

The new modal operator, denoted by $[\![a?]\!]$, can be read as a *forcing* box modality: the action specified in the modality must be possible in order for a process to satisfy the formula. This operator can be described with the classic modalities in HML: $[\![a?]\!]\phi$ is equivalent to $\langle a? \rangle\mathbf{tt} \wedge [\,a?\,]\phi$.

Now with this logic, we can define a preorder $\leq_{\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}}$ in terms of the formulae that the specification satisfies: $s \leq_{\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}} i \quad \text{iff} \quad \forall \phi \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}} \quad (s \models \phi \Rightarrow i \models \phi)$.

We note that the logics $\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$ and $\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ are dual. In fact, there exist mutual transformations between both sets of formulae such that a behaviour satisfies one formula if, and only if, it does *not* satisfy the transformed formula. These statements, which are formalised and proved in Appendix B, are at the heart of the proof of the following result.

Theorem 2. *For all states i, s in some LTS, $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$ iff $s \leq_{\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}} i$.*

4 The relation with a logic for ioco

Input-output conformance (ioco) was introduced by Tretmans in [15]. The intuition behind ioco is that a process i is a correct implementation of a specification s if, for each sequence of actions σ allowed by the specification, all the possible outputs from i after having performed σ are allowed by the specification. This is formalized below in a setting in which all actions are observable.

Definition 7. *Let $(S, I, O, \rightarrow)$ be an LTS with inputs and outputs. We define the traces of a state $p \in S$ as $\mathsf{traces}(p) = \{\sigma \mid \exists p'.\ p \xrightarrow{\sigma} p'\}$. Given a trace σ, we define p after $\sigma = \{p' \mid p' \in S,\ p \xrightarrow{\sigma} p'\}$. For each $T \subseteq S$, we set $\mathsf{Out}(T) = \bigcup_{p \in T} \mathsf{outs}(p)$. Finally, the relation $\mathsf{ioco} \in S \times S$ is defined as:*

$$i \,\mathsf{ioco}\, s \text{ iff } \mathsf{Out}(i \ \mathsf{after}\ \sigma) \subseteq \mathsf{Out}(s \ \mathsf{after}\ \sigma), \text{for all } \sigma \in \mathsf{traces}(s).$$

As shown in [8, Theorem 1], $\mathsf{ioco}\underline{\mathsf{s}}$ is included in ioco.

In the setting of Tretmans' standard ioco theory [15], only input-enabled implementations are considered. A state i in an LTS is input enabled if every state i' that is reachable from i is able to perform every input action, that is, $i' \xrightarrow{a?}$ holds for each $a? \in I$ and for each state i' that is reachable from i.

In [3] Beohar and Mousavi introduced an explicit logical characterization of ioco. This characterization uses a non-standard modal operator reminiscent of our $[\![\cdot]\!]$, denoted by $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$[3]. However, output actions can also be used as labels of $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$. This modality can be extended to traces τ as follows: $p \models \lfloor\!\lfloor \tau \rfloor\!\rfloor \phi$ if, and only if, $p \xrightarrow{\tau}$ and $p' \models \phi$, for each p' such that $p \xrightarrow{\tau} p'$. (Note that, for the particular case of input actions $a?$, the semantics of $\lfloor\!\lfloor a? \rfloor\!\rfloor$ coincides with that of $[\![a?]\!]$.)

The explicit logical characterization of ioco given in [3] is defined by means of two different subclasses of logical formulae. The first subclass permits only formulae of the form $\lfloor\!\lfloor \tau \rfloor\!\rfloor [\,b\,]\mathbf{ff}$, where τ is a trace and b is an output action.

For the second subclass of formulae, Beohar and Mousavi consider an extension of the operator $[\,\cdot\,]$ to traces, defined as: $p \models [\,\tau\,]\phi$ if, and only if, $p' \models \phi$ for each p' such that $p \xrightarrow{\tau} p'$. This second subclass permits only formulae of the form $[\,\tau\,][\,b\,]\mathbf{ff}$, where τ is a trace and b is an output action.

The formulae in each of these two subclasses characterize one defining property of the ioco-relation. This intuition is made precise in the following lemma.

Lemma 1 ([3]). *For each sequence of actions τ, output action b and process p the following statements hold:*

1. *$\tau \in \mathsf{traces}(p)$ and $b \notin \mathsf{Out}(p \ \mathsf{after}\ \tau)$ iff $p \models \lfloor\!\lfloor \tau \rfloor\!\rfloor [\,b\,]\mathbf{ff}$.*
2. *$b \notin \mathsf{Out}(p \ \mathsf{after}\ \tau)$ iff $p \models [\,\tau\,][\,b\,]\mathbf{ff}$.*

The resulting logical characterization theorem for ioco is as follows,

Theorem 3 ([3]). *$i \,\mathsf{ioco}\, s$ iff, for all $\tau \in L^*$, $b \in O$, if $s \models \lfloor\!\lfloor \tau \rfloor\!\rfloor [\,b\,]\mathbf{ff}$, then $i \models [\,\tau\,][\,b\,]\mathbf{ff}$.*

[3] In fact, the symbol used to denote the operator $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$ in [3] is $\langle\!| \cdot |\!\rangle$, but we prefer to use an alternative notation in order to avoid confusion with our modal operator $\langle\!| \cdot |\!\rangle$.

The above result is the counterpart of Theorem 2 in the setting of ioco. Note, however, that Theorem 3 is not a classic modal characterization result (as it is the case of, for example, Theorem 2) where if the implementation i is correct with respect to the specification s and s satisfies a formula, then also i satisfies it. Here the implementation does not need to satisfy the properties that hold for the specification. By way of example, implementations need not exhibit all the traces of a specification they correctly implement.

4.1 Relation with $\mathsf{ioco}\underline{\mathsf{s}}$

Theorem 2 in [13] states that if i is input enabled, $i \ \mathsf{ioco}\ s$ implies $i\ \mathsf{ioco}\underline{\mathsf{s}}\ s$. This means that, when restricted to input-enabled implementations, ioco and $\mathsf{ioco}\underline{\mathsf{s}}$ coincide, and therefore the logics characterizing $\mathsf{ioco}\underline{\mathsf{s}}$ presented in this paper also characterize ioco over that class of LTSs. Unfortunately, however, Theorem 2 in [13] does *not* hold, as shown in the following example.

Example 2. Let s and i be defined as follows, where we assume that $I = \{a?, b?\}$.

$$\begin{tikzcd}
& s \arrow[loop above, "\delta!"] \arrow[dl, "a?"'] \arrow[dr, "a?"] & & & i \arrow[loop above, "\delta!"] \arrow[loop right, "b?"] \arrow[d, "a?"] \\
s_1 \arrow[dr, "b!"'] & & s_2 \arrow[dl, "a!"] & & i' \arrow[loop above, "{a?, b?}"] \arrow[d, bend right, "b!"'] \arrow[d, bend left, "a!"] \\
& \cdot \arrow[loop below, "\delta!"] & & & \cdot \arrow[loop below, "{\delta!, a?, b?}"]
\end{tikzcd}$$

Note that i is input-enabled, as required by the theory of ioco. It is easy to see that $i\ \mathsf{ioco}\ s$. On the other hand, $i \not\mathrel{\mathsf{ioco}\underline{\mathsf{s}}}\ s$ because each $\mathsf{ioco}\underline{\mathsf{s}}$ relation containing the pair (i, s) would also have to contain the pair (i', s_1) or the pair (i', s_2). However, no relation including either of those pairs is an $\mathsf{ioco}\underline{\mathsf{s}}$-relation because $i' \xrightarrow{a!}$ and $i' \xrightarrow{b!}$, but $s_1 \not\xrightarrow{a!}$ and $s_2 \not\xrightarrow{b!}$.

As we will now argue, the logics for ioco and $\mathsf{ioco}\underline{\mathsf{s}}$ are incomparable in terms of their expressive power. First of all, note that, if we consider only input-enabled implementations, the formulae of the form $[\,\tau\,][\,b\,]\mathbf{ff}$, with τ a trace, can be expressed in $\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ since in an input-enabled scenario $\llbracket a? \rrbracket$ has the same semantics as $[\,a?\,]$. On the other hand, it is not possible to define a formula $\phi \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ that captures Lemma 1(1). Indeed, by way of example, consider $\phi = \llfloor x! \rrfloor [\,b!\,]\mathbf{ff}$. Any specification s would have to satisfy ϕ iff $s \xrightarrow{x!}$ and $s' \models [\,b!\,]\mathbf{ff}$, for all $s \xrightarrow{x!} s'$. Now, assume that we have in $\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ a formula ψ whose semantics coincides with that of $\llfloor x! \rrfloor [\,b!\,]\mathbf{ff}$. Let

$$\begin{tikzcd} s \arrow[loop left, "a!"] \arrow[loop right, "x!"] & & i \arrow[loop left, "a!"] \end{tikzcd}$$

It is easy to see that i $\mathsf{ioco\underline{s}}$ s, but $s \models \psi$ and $i \not\models \psi$. In other words, ψ is a formula that distinguishes processes related by $\mathsf{ioco\underline{s}}$. Hence, such a formula ψ cannot be in any logic that characterizes $\mathsf{ioco\underline{s}}$.

On the other hand, let us consider the two processes of Example 2 and the formula $\phi = [\![a?]\!]([\,a!\,]\mathbf{ff} \vee [\,b!\,]\mathbf{ff}) \in \widetilde{\mathcal{L}}_{\mathsf{ioco\underline{s}}}$. As we already stated in Example 2, i ioco s, but $s \models \phi$ and $i \not\models \phi$. Hence, ϕ can distinguish processes that are ioco-related.

5 A rule format for iocos

In this section we study compositionality for $\mathsf{ioco\underline{s}}$ and present a congruence rule format for the input-output conformance simulation preorder based on the GSOS format proposed by Bloom, Istrail and Meyer [5]. The restriction to GSOS rules is partly justified by our wish to have a purely syntactic rule format and by the undecidability results presented in [12]. In what follows, we assume that the reader is familiar with the standard notions of signature and of term over a signature.

We recall that a deduction rule for an operator f of arity n in some signature Σ is in the *GSOS format* if, and only if, it has the following form:

$$\frac{\{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\} \cup \{x_i \stackrel{b_{ik}}{\not\longrightarrow} \mid 1 \leq i \leq n, 1 \leq k \leq \ell_i\}}{f(\boldsymbol{x}) \xrightarrow{a} C[\boldsymbol{x}, \boldsymbol{y}]} \quad (1)$$

where the x_i's and the y_{ij}'s ($1 \leq i \leq n$ and $1 \leq j \leq m_i$) are all distinct variables, m_i and ℓ_i are natural numbers, $C[\boldsymbol{x}, \boldsymbol{y}]$ is a term over Σ with variables including at most the x_i's and y_{ij}'s, and the a_{ij}'s, b_{ik}'s and a are actions from L. The above rule is said to be *f-defining* and *a-emitting.* Its *positive trigger for variable x_i* is the set $\{a_{ij} \mid 1 \leq j \leq m_i\}$ and its *negative trigger for variable x_i* is the set $\{b_{ik} \mid 1 \leq k \leq \ell_i\}$. The *source of the conclusion* of the rule is $f(\boldsymbol{x})$.

A GSOS language is a triple (Σ, L, D) where Σ is a finite signature, L is a finite set of labels and D is a finite set of deduction rules in the GSOS format. In what follows, we assume, without loss of generality, that all f-defining rules have the same source of their conclusions.

A GSOS language naturally defines a set of transitions over the variable-free terms over Σ by structural induction: for vectors of such terms $\boldsymbol{p}$ (with typical entry p_i) and $\boldsymbol{q}$ (with entries $q_{i,j}$), there is a transition $f(\boldsymbol{p}) \xrightarrow{a} C[\boldsymbol{p}, \boldsymbol{q}]$ if, and only if, there is an f-defining rule of the form (1) such that

- $p_i \xrightarrow{a_{ij}} q_{ij}$ for each $1 \leq i \leq n$ and $1 \leq j \leq m_i$ and
- $p_i \stackrel{b_{ik}}{\not\longrightarrow}$ for each $1 \leq i \leq n$ and $1 \leq k \leq \ell_i$.

Note that GSOS rules define operations over states in an arbitrary LTS with inputs and outputs. In what follows, we apply derived operations built over the signature of a GSOS language to states in the collection of LTSs with input and output actions.

Definition 8. *An operation f in a GSOS language is in* $\underline{\mathsf{iocos}}$-format *if the collection of f-defining rules satisfies the following conditions:*

1. *Each $a?$-emitting rule, where $a?$ is an input action, has only output actions as labels of negative premises and input actions as labels of positive premises.*
2. *For each input action $a?$ and each pair of rules $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a?} t}$ and $r' = \frac{H'}{f(x_1,\ldots,x_n) \xrightarrow{a?} t'}$, there is a rule $r'' = \frac{H''}{f(x_1,\ldots,x_n) \xrightarrow{a?} t'}$ such that*
   (a) *for each $1 \le i \le n$, the positive trigger for variable x_i in r'' is included in the positive trigger for variable x_i in r;*
   (b) *for each $1 \le i \le n$, the negative trigger for variable x_i in r'' is included in the negative trigger for variable x_i in r;*
   (c) *if $x_i \xrightarrow{b?} z$ is contained in H'' and z occurs in t', then $x_i \xrightarrow{b?} z$ is also contained in H'.*
3. *Each $a!$-emitting rule, where $a!$ is an output action, has only input actions as labels of negative premises and output actions as labels of positive premises.*

A GSOS language is in $\underline{\mathsf{iocos}}$-format *if so is each of its operations.*

Theorem 4. $\underline{\mathsf{iocos}}$ *is a precongruence for each GSOS language in* $\underline{\mathsf{iocos}}$ *format.*

The proof of this result may be found in Appendix C.

As an example of application of the above result, we show that the merge operator from [2] can be expressed in our rule format.

Example 3. Merge, or conjunction, is a composition operator from the theory of ioco. It acts as a logical conjunction of requirements, that is, it describes systems by a conjunction of sub-systems, or sub-specifications. We denote by $\bigwedge_{i=1}^{n} s_i$ the result of the merge of the states s_i, with $1 \le i \le n$. In [2] is noted that, in general, the merge of two systems can lead to invalid states (for example the merge of a quiescent state with another with some output). The solution is to add a pruning algorithm after calculating the merge. Here we just show the merge operator and not that pruning algorithm. (See also Example 4.)

The merge operator can be formalized using the following GSOS rules (one such rule for each $a \in L$):

$$\frac{\{x_i \xrightarrow{a} y_i \mid 1 \le i \le n\}}{\bigwedge_{i=1}^{n} x_i \xrightarrow{a} \bigwedge_{i=1}^{n} y_i} \,.$$

It is immediate to check that the above rules are in $\underline{\mathsf{iocos}}$-format. Therefore the above theorem yields that the merge operator preserves $\underline{\mathsf{iocos}}$.

In Appendix D, we present some examples showing that the restrictions of the rule format from Definition 8 cannot be relaxed easily.

5.1 A rule format for coherent quiescent behaviour

Operators for constructing LTSs with inputs and outputs should ensure 'coherent quiescent behaviour' in the sense of Definition 1. This means that each operator f, when applied to a vector of states $\boldsymbol{p}$ in an LTS, should satisfy the following property:

$$f(\boldsymbol{p}) \xrightarrow{\delta!} p' \text{ iff } p' = f(\boldsymbol{p}) \text{ and, for each } a! \in O \setminus \{\delta!\},\ f(\boldsymbol{p}) \overset{a!}{\not\longrightarrow} . \tag{2}$$

In what follows, we will isolate sufficient conditions on the GSOS rules that define f that guarantee the above-mentioned property.

Definition 9. *We say that the following sets of formulae contradict each other:*

- $\{x \xrightarrow{a} y\}$ *and* $\{x \overset{a}{\not\longrightarrow}\}$ *for* $a \in L$,
- $\{x \xrightarrow{b!} y\}$ *and* $\{x \xrightarrow{\delta!} z\}$ *for* $b! \in O \setminus \{\delta!\}$, *and*
- H *and* H' *when* H *and* H' *are non-empty and* $H \cup H' = \{x \overset{b!}{\not\longrightarrow} \mid b! \in O\}$.

Formulae $x \xrightarrow{a} y$ *and* $x \overset{a}{\not\longrightarrow}$ *are said to negate each other.*

We say that two sets of formulae H_1 *and* H_2 *are* contradictory *if there are* $H_1' \subseteq H_1$ *and* $H_2' \subseteq H_2$ *such that* H_1' *and* H_2' *contradict each other.*

Intuitively, two sets of contradictory formulae cannot be both satisfied by states in an LTS. For example, in the light of the requirement on quiescent behaviour in Definition 1, there is no state p in an LTS such that $p \overset{b!}{\not\longrightarrow}$ for each $b! \in O$. This observation motivates the third requirement in Definition 9.

Definition 10. *We say that an operation* f *is* quiescent consistent *if the set of rules for* f *satisfies the following two constraints:*

[δ_1] *If* $H/f(\boldsymbol{x}) \xrightarrow{\delta!} t$ *is a rule for* f *then*

1. *for each* f*-defining rule* $H'/f(\boldsymbol{x}) \xrightarrow{b!} t'$ *with* $b! \in O \setminus \{\delta!\}$, *the sets* H *and* H' *are contradictory, and*
2. $t = f(\boldsymbol{y})$ *for some vector of variables* $\boldsymbol{y}$ *such that, for each index* i, *either* $y_i = x_i$ *or* $x_i \xrightarrow{\delta!} y_i \in H$.

[δ_2] *Let* $\{r_1, \ldots r_n\}$ *be the set of output-emitting rules for* f *not having* $\delta!$ *as label of their conclusions. Then the set of rules for* f *contains all rules of the form* $\{l_1, \ldots, l_n\}/f(\boldsymbol{x}) \xrightarrow{\delta!} f(\boldsymbol{x})$, *where* l_i *negates some premise of* r_i *and no two sets of formulae included in* $\{l_1, \ldots l_n\}$ *contradict each other.*

A GSOS language is quiescent consistent *if so is each operation in it.*

Theorem 5. *If* f *is quiescent consistent then Equation 2 holds for* f.

The proof of this result may be found in Appendix E.

Example 4. Consider the merge, or conjunction, operator from [2] described in Example 3. As remarked in [2, Example 2], the merge operator may produce an invalid LTS when applied to states from an LTS. Note that the set of rules for the n-ary merge operator satisfy constraint $[\delta_1]$ in Definition 10, but not constraint $[\delta_2]$. Constraint $[\delta_2]$ also suggests how to add rules to those of the merge operator so that it preserves consistent quiescent behaviour. By way of example, consider the binary version of the merge operator, and assume that $a!$ and $b!$ are the only two output actions different from $\delta!$. Then one should add the following four rules to those for the binary merge given in Example 3:

$$\frac{\{x_1 \xrightarrow{\alpha!} y_1,\ x_2 \xrightarrow{\beta!} y_2\}}{x_1 \wedge x_2 \xrightarrow{\delta!} x_1 \wedge x_2}\ (\alpha \neq \beta)\ .$$

The resulting operation is quiescent consistent and, by Theorem 5, satisfies Equation 2.

6 Conclusion

In this paper, we have developed the theory of $\mathsf{ioco}\underline{\mathsf{s}}$ [8–10] by studying logical characterisations of this relation and its compositionality. We have also compared the proposed logical characterisation of $\mathsf{ioco}\underline{\mathsf{s}}$ with an existing logical characterisation for ioco proposed by Behoar and Mousavi. The article also offers a precongruence rule format for $\mathsf{ioco}\underline{\mathsf{s}}$ and a rule format ensuring that operations take quiescence properly into account. Both rule formats are based on the GSOS format by Bloom, Istrail and Meyer.

Avenues for future research we are currently pursuing include an extension of the logic for $\mathsf{ioco}\underline{\mathsf{s}}$ with fixed points, a characteristic formula construction for finite-state behaviours with respect to $\mathsf{ioco}\underline{\mathsf{s}}$, an application of the divide and congruence approach from [7] to the definition of a congruence rule format for $\mathsf{ioco}\underline{\mathsf{s}}$ (as done in [4] for the XY-simulation preorder) and a compositionality result for the logic characterising $\mathsf{ioco}\underline{\mathsf{s}}$ over languages in $\mathsf{ioco}\underline{\mathsf{s}}$ format.

References

1. Luca Aceto, Wan Fokkink, and Chris Verhoef. *Handbook of Process Algebra*, chapter Structural operational semantics, pages 197–292. Elsevier, 2001.
2. Nikola Benes, Przemyslaw Daca, Thomas A. Henzinger, Jan Kretínský, and Dejan Nickovic. Complete composition operators for ioco-testing theory. In *Proceedings of the 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering, CBSE 2015*, pages 101–110, 2015.
3. Harsh Beohar and Mohammad Reza Mousavi. Two logical characterizations for input-output conformance. In Preproceedings of EXPRESS/SOS'14 (Short paper), July 2014.
4. Harsh Beohar and Mohammad Reza Mousavi. A pre-congruence format for XY-simulation. In *Fundamentals of Software Engineering - 6th International Conference, FSEN 2015 Tehran, Iran, April 22–24, 2015, Revised Selected Papers*, volume 9392 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2015.

5. Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
6. David de Frutos-Escrig, Carlos Gregorio-Rodríguez, Miguel Palomino, and David Romero-Hernández. Unifying the linear time-branching time spectrum of process semantics. *Logical Methods in Computer Science*, 9(2:11):1–74, 2013.
7. Wan Fokkink, Rob J. van Glabbeek, and Paulien de Wind. Compositionality of Hennessy-Milner logic by structural operational semantics. *Theor. Comput. Sci.*, 354(3):421–440, 2006.
8. Carlos Gregorio-Rodríguez, Luis Llana, and Rafael Martínez-Torres. Input-output conformance simulation (iocos) for model based testing. In Dirk Beyer and Michele Boreale, editors, *FMOODS/FORTE*, volume 7892 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2013.
9. Carlos Gregorio-Rodríguez, Luis Llana, and Rafael Martínez-Torres. Effectiveness for input output conformance simulation iocos. In Erika Ábrahám and Catuscia Palamidessi, editors, *FORTE*, volume 8461 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2014.
10. Carlos Gregorio-Rodríguez, Luis Llana, and Rafael Martínez-Torres. Extending mCRL2 with ready simulation and iocos input-output conformance simulation. In Roger L. Wainwright, Juan Manuel Corchado, Alessio Bechini, and Jiman Hong, editors, *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 1781–1788. ACM, 2015.
11. Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.
12. Bartek Klin and Beata Nachyła. Some undecidable properties of SOS specification. To appear.
13. Luis Llana and Rafael Martínez-Torres. IOCO as a simulation. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, Madrid, Spain, September 23-24, 2013, Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2013.
14. Gerjan Stokkink, Mark Timmer, and Mariëlle Stoelinga. Talking quiescence: a rigorous theory that supports parallel composition, action hiding and determinisation. In Alexander K. Petrenko and Holger Schlingloff, editors, *MBT*, volume 80 of *EPTCS*, pages 73–87, 2012.
15. Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.
16. Jan Tretmans. Model based testing with labelled transition systems. In Robert M. Hierons, Jonathan P. Bowen, and Mark Harman, editors, *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.
17. Machiel van der Bijl, Arend Rensink, and Jan Tretmans. Compositional testing with ioco. In *Formal Approaches to Software Testing, Third International Workshop on Formal Approaches to Testing of Software, FATES 2003, Montreal, Quebec, Canada, October 6th, 2003*, volume 2931 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2003.
18. Rob J. van Glabbeek. *Handbook of Process Algebra*, chapter The Linear Time – Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes, pages 3–99. Elsevier, 2001.

A Proof of Theorem 1

We prove the two implications separately.

- 'ONLY IF IMPLICATION': Assume that i $\mathsf{ioco}\underline{\mathsf{s}}$ s and $i \models \phi$ with $\phi \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$. We show that $s \models \phi$ by structural induction over the formula ϕ. We limit ourselves to presenting the case that $\phi = \langle\!| a? |\!\rangle\varphi$, for some φ. Since $s \models \langle\!| a? |\!\rangle\varphi$ holds when $s \overset{a?}{\not\longrightarrow}$, in what follows we may assume that $a? \in \mathsf{ins}(s)$. Since i $\mathsf{ioco}\underline{\mathsf{s}}$ s, by Definition 2.1, $\mathsf{ins}(s) \subseteq \mathsf{ins}(i)$. So, from $i \models \langle\!| a? |\!\rangle\varphi$ we obtain that there exists some i' such that $i \xrightarrow{a?} i'$ and $i' \models \varphi$. By Definition 2.2, we have that there exists some s' such that $s \xrightarrow{a?} s'$ such that i' $\mathsf{ioco}\underline{\mathsf{s}}$ s'. Now, since $i' \models \varphi$, by the induction hypothesis, we have also that $s' \models \varphi$. That is, $s \models \langle\!| a? |\!\rangle\varphi$, which was to be shown.
- 'IF IMPLICATION': Consider the relation $R = \{(i, s) \mid i \leq_{\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}} s\}$. We claim that R is an $\mathsf{ioco}\underline{\mathsf{s}}$-relation. We will prove that claim by contradiction. Assume that $(i, s) \in R$ does not satisfy the requirements in Definition 2. We will see that there exists a formula φ in $\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$ such that $i \models \varphi$ and $s \not\models \varphi$, contradicting the assumption that $i \leq_{\mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}} s$. We distinguish three cases according to the conditions in Definition 2.
  - Assume that $\mathsf{ins}(s) \not\subseteq \mathsf{ins}(i)$. Then, there exists some $a? \in \mathsf{ins}(s) \setminus \mathsf{ins}(i)$. In this case, $i \models \langle\!| a? |\!\rangle\mathbf{ff}$ but $s \not\models \langle\!| a? |\!\rangle\mathbf{ff}$.
  - Assume that there exist $a? \in \mathsf{ins}(s) \cap \mathsf{ins}(i)$ and i' such that $i \xrightarrow{a?} i'$ and $(i', s') \notin R$ for each s' such that $s \xrightarrow{a?} s'$. Since for each $s \xrightarrow{a?} s'$ we have $(i', s') \notin R$, there exist formulae $\varphi_{s'}$ such that $i' \models \varphi_{s'}$ and $s' \not\models \varphi_{s'}$. Let $\varphi = \langle\!| a? |\!\rangle \bigwedge_{s \xrightarrow{a?} s'} \varphi_{s'}$. By construction $i \models \varphi$ and $s \not\models \varphi$.
  - Assume that there exist $o! \in O$ and i' such that $i \xrightarrow{o!} i'$ and $(i', s') \notin R$ for all $s \xrightarrow{o!} s'$. As above, for each $s \xrightarrow{o!} s'$ there exists some formula $\varphi_{s'}$ such that $i' \models \varphi_{s'}$ and $s' \not\models \varphi_{s'}$. In this case, the formula $\varphi = \langle o! \rangle \bigwedge_{s \xrightarrow{o!} s'} \varphi_{s'}$ is such that $i \models \varphi$ but $s \not\models \varphi$. □

B Proof of Theorem 2

Definition 11. *We define the bijection $\mathcal{T} : \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}} \to \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ by induction on the structure of formulae in the following way:*

- $\mathcal{T}(\mathbf{tt}) = \mathbf{ff}$.
- $\mathcal{T}(\mathbf{ff}) = \mathbf{tt}$.
- $\mathcal{T}(\phi_1 \wedge \phi_2) = \mathcal{T}(\phi_1) \vee \mathcal{T}(\phi_2)$.
- $\mathcal{T}(\phi_1 \vee \phi_2) = \mathcal{T}(\phi_1) \wedge \mathcal{T}(\phi_2)$.
- $\mathcal{T}(\langle\!| a? |\!\rangle\phi) = [\![a?]\!]\mathcal{T}(\phi)$.
- $\mathcal{T}(\langle x! \rangle\phi) = [\, x! \,]\mathcal{T}(\phi)$.

The inverse function $\mathcal{T}^{-1} : \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}} \to \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$ is defined in the obvious way.

In order to prove Theorem 2, we first prove the following lemma to the effect that a behaviour satisfies some formula ϕ if, and only if, it does not satisfy the transformed formula $\mathcal{T}(\phi)$.

Lemma 2. *For each state p in some LTS, and formula $\phi \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$, we have that*

(i) if $p \models \phi$ then $p \not\models \mathcal{T}(\phi)$, and
(ii) if $p \not\models \phi$ then $p \models \mathcal{T}(\phi)$.

Proof. We prove both statements by structural induction over the formula $\phi \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$. We only show the case for a formula $\phi = \langle\!| a? |\!\rangle \varphi$.

(i): By definition $\mathcal{T}(\langle\!| a? |\!\rangle \varphi) = [\![a?]\!] \mathcal{T}(\varphi)$. Consider some process p such that $p \models \langle\!| a? |\!\rangle \varphi$. Since $p \models \langle\!| a? |\!\rangle \varphi$ we have that either $p \overset{a?}{\not\longrightarrow}$, or there exists $p \xrightarrow{a?} p'$ such that $p' \models \varphi$. If $p \overset{a?}{\not\longrightarrow}$, then $p \not\models [\![a?]\!] \mathcal{T}(\varphi)$. Assume now that there exists some p' such that $p \xrightarrow{a?} p'$ and $p' \models \varphi$. By the inductive hypothesis, $p' \not\models \mathcal{T}(\varphi)$. Therefore $p \not\models [\![a?]\!] \mathcal{T}(\varphi)$, and we are done.

(ii): By definition $\mathcal{T}(\langle\!| a? |\!\rangle \varphi) = [\![a?]\!] \mathcal{T}(\varphi)$. Consider some process p such that $p \not\models \phi$. This means that $p \xrightarrow{a?}$ and $p' \not\models \varphi$ for all $p \xrightarrow{a?} p'$. By the induction hypothesis, $p' \models \mathcal{T}(\varphi)$ for each p' such that $p \xrightarrow{a?} p'$. Since $p \xrightarrow{a?}$, we obtain that $p \models [\![a?]\!] \mathcal{T}(\varphi)$, thus concluding the proof. $\square$

The following corollary will be useful in the proof of Theorem 2.

Corollary 1. *For every state p in some LTS, and all formulae $\phi \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$ and $\psi \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$, the following properties are satisfied:*

(i) $p \models \phi$ iff $p \not\models \mathcal{T}(\phi)$.
(ii) $p \models \psi$ iff $p \not\models \mathcal{T}^{-1}(\psi)$.

Finally, we have all we need in order to prove Theorem 2, which is the equivalent result to Theorem 1 for $\widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$.

Proof. Assume that $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$ and let us consider $\phi \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ such that $s \models \phi$. By Corollary 1.(ii), $s \not\models \mathcal{T}^{-1}(\phi)$, with $\mathcal{T}^{-1}(\phi) \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$. Now, by Theorem 1, this implies $i \not\models \mathcal{T}^{-1}(\phi)$ and by Lemma 2.ii, $i \models \mathcal{T}(\mathcal{T}^{-1}(\phi)) = \phi$.

On the other hand, let us suppose that for all $\phi \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$, if $s \models \phi$ then $i \models \phi$. By Theorem 1 to show that $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$ it suffices to prove that $i \models \varphi$ implies $s \models \varphi$ for each $\varphi \in \mathcal{L}_{\mathsf{ioco}\underline{\mathsf{s}}}$. To this end let us suppose that $i \models \varphi$. By Corollary 1.(i), we have that $i \not\models \mathcal{T}(\varphi)$. Now, by hypothesis, since $\mathcal{T}(\varphi) \in \widetilde{\mathcal{L}}_{\mathsf{ioco}\underline{\mathsf{s}}}$ it must also be the case that $s \not\models \mathcal{T}(\varphi)$. Applying again Corollary 1.(i) we obtain that $s \models \varphi$. Thus, we obtain $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$, finishing the proof. $\square$

C Proof of Theorem 4

Remark 2. All the substitutions we consider from now on map variables either to variable-free terms or to states in some LTS.

Let $\mathcal{S}$ be the least binary relation that includes $\mathsf{ioco\underline{s}}$ and such that: if $p_i \,\mathcal{S}\, q_i$, for each $1 \leq i \leq n$, and f is an n-ary operation then $f(p_1, \ldots, p_n) \,\mathcal{S}\, f(q_1, \ldots, q_n)$.

The following property of $\mathcal{S}$ will be useful in what follows.

Lemma 3. *Let t be a term. Assume that σ and ρ are two substitutions such that $\sigma(x) \,\mathcal{S}\, \rho(x)$ for each variable x occurring in t. Then $\sigma(t) \,\mathcal{S}\, \rho(t)$.*

We show that $\mathcal{S}$ is an $\mathsf{ioco\underline{s}}$ simulation by induction on the definition of $\mathcal{S}$. This is clear if $p \,\mathcal{S}\, q$ because $p \;\mathsf{ioco\underline{s}}\; q$. Assume therefore that

$$p = f(p_1, \ldots, p_n) \,\mathcal{S}\, f(q_1, \ldots, q_n) = q$$

because $p_i \,\mathcal{S}\, q_i$ for each $1 \leq i \leq n$. We proceed to prove that each of the defining conditions for iocos simulations holds for p and q. In doing so, we shall use, as our inductive hypothesis, the fact that those conditions hold for $p_i \,\mathcal{S}\, q_i$ $(1 \leq i \leq n)$. In particular, we have that $\mathsf{ins}(q_i) \subseteq \mathsf{ins}(p_i)$ $(1 \leq i \leq n)$.

– We prove, first of all, that the set of initial input actions of q is included in the set of initial input actions of p. To this end, assume that $a?$ is an input action and $q = f(q_1, \ldots, q_n) \xrightarrow{a?}$. We shall prove that $p = f(p_1, \ldots, p_n) \xrightarrow{a?}$ also holds.
  Since $q \xrightarrow{a?}$, there are a rule r of the form (1) on page 8 and a substitution σ such that
  - $\sigma(x_i) = q_i$ for each $1 \leq i \leq n$,
  - $q_i \xrightarrow{a_{ij}} \sigma(y_{ij})$ for each $1 \leq i \leq n$ and $1 \leq j \leq m_i$ and
  - $q_i \stackrel{b_{ik}}{\not\longrightarrow}$ for each $1 \leq i \leq n$ and $1 \leq k \leq \ell_i$.

  As $a?$ is an input action, by requirement 1 in Definition 8, we have that each a_{ij} is an input action and each b_{ik} is an output action. Since $p_i \,\mathcal{S}\, q_i$ $(1 \leq i \leq n)$, by the inductive hypothesis, we therefore have that:
  - for each $1 \leq i \leq n$ and $1 \leq j \leq m_i$, as $a_{ij} \in \mathsf{ins}(q_i) \subseteq \mathsf{ins}(p_i)$, there is some state p_{ij} such that $p_i \xrightarrow{a_{ij}} p_{ij}$ and
  - $p_i \stackrel{b_{ik}}{\not\longrightarrow}$, for each $1 \leq i \leq n$ and $1 \leq k \leq \ell_i$.

  Therefore rule r can be used to infer that $p \xrightarrow{a?}$, and we are done.
– Assume now that $p = f(p_1, \ldots, p_n) \xrightarrow{a?} p'$ and $a?$ is an initial input action of $q = f(q_1, \ldots, q_n)$. As $a?$ is an initial input action of $q = f(q_1, \ldots, q_n)$, there are a rule $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a?} t}$ of the form (1) on page 8 and a substitution σ such that
  - $\sigma(x_i) = q_i$ for each $1 \leq i \leq n$, and
  - 'σ satisfies all the premises in H'.

  Moreover, there are a rule $r' = \frac{H'}{f(x_1,\ldots,x_n) \xrightarrow{a?} t'}$ of the form (1) on page 8 and a substitution σ' such that
  - $\sigma'(x_i) = p_i$ for each $1 \leq i \leq n$,
  - 'σ' satisfies all the premises in H'', and
  - $\sigma'(t') = p'$.

Our goal is to prove that $q = f(q_1, \ldots, q_n) \xrightarrow{a?} q'$ for some q' such that $p' \mathcal{S} q'$. To this end, observe, first of all, that requirement 2 in Definition 8 tells us that there is an f-defining rule $r'' = \frac{H''}{f(x_1,\ldots,x_n) \xrightarrow{a?} t'}$ such that

- for each $1 \leq i \leq n$, the positive trigger for variable x_i in r'' is included in the positive trigger for variable x_i in r;
- for each $1 \leq i \leq n$, the negative trigger for variable x_i in r'' is included in the negative trigger for variable x_i in r;
- if $x_i \xrightarrow{b?} z$ is contained in H'' and z occurs in t', then $x_i \xrightarrow{b?} z$ is also contained in H'.

To complete the proof for this case, we will now show how to use the rule r'' to prove the required transition $q = f(q_1, \ldots, q_n) \xrightarrow{a?} q'$. To this end, we will construct a substitution ρ with the following properties:

1. $\rho(x_i) = q_i$ for each $1 \leq i \leq n$,
2. $\sigma'(z) \, \mathcal{S} \, \rho(z)$ for each variable z occurring in t' and
3. 'ρ satisfies all the premises in H'''.

The first condition above gives us that $\rho(f(x_1, \ldots, x_n)) = q$. The second yields that $p' = \sigma'(t') \, \mathcal{S} \, \rho(t')$ by Lemma 3. From the third, we obtain that

$$q = f(q_1, \ldots, q_n) \xrightarrow{a?} \rho(t').$$

Therefore, the substitution ρ and the rule r'' prove the existence of the required transition $q = f(q_1, \ldots, q_n) \xrightarrow{a?} \rho(t') = q'$ with $p' \, \mathcal{S} \, q'$.
To meet the first condition above, we start by setting $\rho(x_i) = q_i$ for each $1 \leq i \leq n$. Note, next, that, since the negative trigger for variable x_i in r'' is included in the negative trigger for variable x_i in r $(1 \leq i \leq n)$, and σ satisfies H, any substitution ρ such that $\rho(x_i) = q_i$ $(1 \leq i \leq n)$ meets all the negative premises in H''. Our aim now is to extend the definition of ρ to all the other variables occurring in rule r'' in such a way that the other two above-mentioned requirements are met. To this end, consider a positive premise $x_i \xrightarrow{b?} z \in H''$. We know that the positive trigger for variable x_i in r'' is included in the positive trigger for variable x_i in r. Therefore H contains a positive premise $x_i \xrightarrow{b?} w$ for some variable w. As σ satisfies H, we have that $\sigma(x_i) = q_i \xrightarrow{b?} \sigma(w)$ and therefore $b? \in \mathsf{ins}(q_i)$. If z does not occur in t', setting $\rho(z) = \sigma(w)$ will satisfy the premise $x_i \xrightarrow{b?} z \in H''$. Assume now that z does occur in t'. In this case, by requirement 2c in the definition of the rule format (Definition 8), we know that $x_i \xrightarrow{b?} z$ is also contained in H'. Since σ' satisfies H', we have that

$$\sigma'(x_i) = p_i \xrightarrow{b?} \sigma'(z).$$

As $p_i \, \mathcal{S} \, q_i$, $b? \in \mathsf{ins}(q_i)$ and $\sigma'(x_i) = p_i \xrightarrow{b?} \sigma'(z)$, the inductive hypothesis yields that

$$\rho(x_i) = q_i \xrightarrow{b?} q_i' \text{ for some } q_i' \text{ such that } \sigma'(z) \, \mathcal{S} \, q_i'.$$

We can therefore set $\rho(z) = q'_i$ in order to keep meeting the last two requirements on ρ.
Continuing in this fashion until we have exhausted all the positive premises in H'' completes the proof for this case.

– Assume that $p = f(p_1, \ldots, p_n) \xrightarrow{a!} p'$ for some output action $a!$ and state p'. Then there are a rule r of the form (1) on page 8 and a substitution σ such that
  - $\sigma(x_i) = p_i$ for each $1 \leq i \leq n$,
  - $p_i \xrightarrow{a_{ij}} \sigma(y_{ij})$ for each $1 \leq i \leq n$ and $1 \leq j \leq m_i$,
  - $p_i \overset{b_{ik}}{\not\longrightarrow}$ for each $1 \leq i \leq n$ and $1 \leq k \leq \ell_i$, and
  - $\sigma(C[\boldsymbol{x}, \boldsymbol{y}]) = p'$.

  Our goal is to use rule r to prove that $q = f(q_1, \ldots, q_n) \xrightarrow{a!} q'$ for some q' such that $p' \,\mathcal{S}\, q'$.
  Since $a!$ is an output action, condition 3 in Definition 8 tell us that each a_{ij} is an output action, and each b_{ik} is an input action. As each a_{ij} is an output action, $p_i \xrightarrow{a_{ij}!} \sigma(y_{ij})$ and $p_i \,\mathcal{S}\, q_i$ $(1 \leq i \leq n)$, the inductive hypothesis yields that for each i and j there is some q_{ij} such that $q_i \xrightarrow{a_{ij}!} q_{ij}$ and $\sigma(y_{ij}) \,\mathcal{S}\, q_{ij}$.
  Note, moreover, that the q_i's satisfy the negative premises of rule r. Indeed, $p_i \overset{b_{ik}}{\not\longrightarrow}$ $(1 \leq i \leq n$ and $1 \leq k \leq \ell_i)$ and, as $p_i \,\mathcal{S}\, q_i$ $(1 \leq i \leq n)$, the inductive hypothesis gives us that the set of input actions of each q_i is included in that of p_i. Therefore, we have that rule r instantiated with a substitution ρ such that

$$\rho(x_i) = q_i$$
$$\rho(y_{ij}) = q_{ij}$$

  yields the transition

$$q = f(q_1, \ldots, q_n) \xrightarrow{a!} \rho(C[\boldsymbol{x}, \boldsymbol{y}]).$$

  By construction, $\sigma(z) \,\mathcal{S}\, \rho(z)$ for each variable z occurring in $C[\boldsymbol{x}, \boldsymbol{y}]$. Therefore Lemma 3 now yields that

$$p' = \sigma(C[\boldsymbol{x}, \boldsymbol{y}]) \,\mathcal{S}\, \rho(C[\boldsymbol{x}, \boldsymbol{y}]),$$

  and we are done. □

D Counterexamples

This appendix presents some examples showing that the restrictions of the rule format from Definition 8 cannot be relaxed easily.

Example 5. This example indicates that the use of input actions in negative premises of input-emitting rules would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{}{f(x) \xrightarrow{\delta!} f(x)} \qquad\qquad \frac{x \overset{a?}{\nrightarrow}}{f(x) \xrightarrow{a?} f(x)}\ ,$$

where $a? \in I$. Now, let us consider the following processes from Example 1:

$$\begin{tikzcd} i \arrow[loop left, "\delta!"] \arrow[loop right, "a?"] & & s \arrow[loop right, "\delta!"] \end{tikzcd}$$

As remarked in Example 1, $i\ \mathsf{ioco}\underline{\mathsf{s}}\ s$. On the other hand, $f(i)\ \mathsf{io}\not{\mathsf{c}}\mathsf{o}\underline{\mathsf{s}}\ f(s)$ because $a? \in \mathsf{ins}(f(s))$ but $a? \notin \mathsf{ins}(f(i))$.

Example 6. This example indicates that the use of output actions in positive premises of input-emitting rules would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{}{f(x) \xrightarrow{\delta!} f(x)} \qquad\qquad \frac{x \xrightarrow{b!} y}{f(x) \xrightarrow{a?} f(x)}\ ,$$

where $b! \in O$. Now, let us consider the following processes:

$$\begin{tikzcd} p \arrow[loop left, "c!"] & & q \arrow[loop left, "b!"] \arrow[loop right, "c!"] \end{tikzcd}$$

where $c! \in O$. We have that $p\ \mathsf{ioco}\underline{\mathsf{s}}\ q$ but $f(p)\ \mathsf{io}\not{\mathsf{c}}\mathsf{o}\underline{\mathsf{s}}\ f(q)$. Indeed, $a? \in \mathsf{ins}(f(q))$ but $a? \notin \mathsf{ins}(f(p))$.

Example 7. This example indicates that not meeting requirement 2a in Definition 8 would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{}{f(x) \xrightarrow{\delta!} f(x)} \qquad \frac{}{0 \xrightarrow{\delta!} 0} \qquad \frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} 0} \qquad \frac{}{f(x) \xrightarrow{a?} f(x)}\ .$$

where $a? \in I$. The above set of rules does not meet requirement 2a in Definition 8. To see this, take

$$\frac{}{f(x) \xrightarrow{a?} f(x)}$$

as rule r and

$$\frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} 0}$$

as rule r'. Note that the only possible choice for rule r'' is r' itself. However, with this choice, the positive trigger for variable x in r'' is $\{a?\}$, which is not included in the positive trigger for variable x in r, which is the empty set.

Now, let us consider again the two processes of Example 5:

$$\begin{tikzcd} i \arrow[loop left, "\delta!"] \arrow[loop right, "a?"] \end{tikzcd} \qquad \begin{tikzcd} s \arrow[loop right, "\delta!"] \end{tikzcd}$$

We know that $i \;\mathsf{ioco}\underline{\mathsf{s}}\; s$ but $f(i) \;\mathsf{io\not{c}o}\underline{\mathsf{s}}\; f(s)$. Indeed, $f(i) \xrightarrow{a?} 0$ and the only way that $f(s)$ can match an $a?$-transition is by $f(s) \xrightarrow{a?} f(s)$. Since $0 \;\mathsf{io\not{c}o}\underline{\mathsf{s}}\; f(s)$ (because $a? \notin \mathsf{ins}(0)$), this implies that $f(i) \;\mathsf{io\not{c}o}\underline{\mathsf{s}}\; f(s)$.

Note that requirements 2b and 2c in Definition 8 are instead met by taking $r' = r''$.

Example 8. This example indicates that not meeting requirement 2b in Definition 8 would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{}{f(x) \xrightarrow{\delta!} f(x)} \qquad \frac{}{0 \xrightarrow{\delta!} 0} \qquad \frac{x \overset{b!}{\not\longrightarrow}}{f(x) \xrightarrow{a?} 0} \qquad \frac{}{f(x) \xrightarrow{a?} f(x)}\ ,$$

where $a? \in I$ and $b! \in O$. The above set of rules does not meet requirement 2b in Definition 8. To see this, take

$$\frac{}{f(x) \xrightarrow{a?} f(x)}$$

as rule r and

$$\frac{x \overset{b!}{\not\longrightarrow}}{f(x) \xrightarrow{a?} 0}$$

as rule r'. Note that the only possible choice for rule r'' is r' itself. However, with this choice, the negative trigger for variable x in r'' is $\{b!\}$, which is not included in the negative trigger for variable x in r, which is the empty set.

Now, let us consider the following processes:

$$\begin{tikzcd} p \arrow[loop left, "c!"] \arrow[loop right, "a?"] \end{tikzcd} \qquad \begin{tikzcd} q \arrow[loop left, "b!"] \arrow[loop right, "c!"] \end{tikzcd}$$

where $c! \in O$. Again, $p \;\mathsf{ioco}\underline{\mathsf{s}}\; q$ but $f(p) \;\mathsf{io\not{c}o}\underline{\mathsf{s}}\; f(q)$. Indeed, $f(p) \xrightarrow{a?} 0$ and the only way $f(q)$ can match an $a?$-transition is by $f(q) \xrightarrow{a?} f(q)$. However, as in the previous case, $0 \;\mathsf{io\not{c}o}\underline{\mathsf{s}}\; f(q)$.

Note that requirements 2a and 2c in Definition 8 are instead met by taking $r' = r''$.

Example 9. This example indicates that not meeting requirement 2c in Definition 8 would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{}{f(x) \xrightarrow{\delta!} f(x)} \qquad \frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} y} \qquad \frac{x \xrightarrow{b?} y}{f(x) \xrightarrow{a?} y} \qquad \frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} f(x)}\ ,$$

where $a?, b? \in I$. The above set of rules does not meet requirement 2c in Definition 8. To see this, take

$$\frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} f(x)}$$

as rule r and

$$\frac{x \xrightarrow{b?} y}{f(x) \xrightarrow{a?} y}$$

as rule r'. Note that the only possible choice for rule r'' meeting requirement 2a and having y as target of its conclusion is

$$\frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} y} \; .$$

However, with this choice, the positive premise $x \xrightarrow{a?} y$ of r'' is not a positive premise of r'.

Now, let us consider the following two processes p and q:

$$\begin{tikzcd} p \arrow[loop left, "{a?,\delta!}"] \arrow[r, "b?"] & 0 \arrow[loop right, "\delta!"] \end{tikzcd} \qquad \begin{tikzcd} q \arrow[loop right, "{a?,\delta!}"] \end{tikzcd}$$

It is easy to see that $p \mathrel{\mathsf{ioco\underline{s}}} q$. However, $f(p) \mathrel{\not\mathsf{ioco\underline{s}}} f(q)$. To see this, observe that $f(q)$ can perform the input action $a?$ and $f(p) \xrightarrow{a?} 0$ by rule r'. The only two possible matching transitions from $f(q)$ are $f(q) \xrightarrow{a?} f(q)$ (using rule r) and $f(q) \xrightarrow{a?} q$ (using rule r''). Since $0 \mathrel{\not\mathsf{ioco\underline{s}}} f(q)$ and $0 \mathrel{\not\mathsf{ioco\underline{s}}} q$ (because $a? \notin \mathsf{ins}(0)$), this implies that $f(p) \mathrel{\not\mathsf{ioco\underline{s}}} f(q)$.

Example 10. This example indicates that the use of output actions in negative premises of output-emitting rules would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{x \xrightarrow{a!} y}{f(x) \xrightarrow{\delta!} f(x)} \qquad\qquad \frac{x \not\xrightarrow{a!}}{f(x) \xrightarrow{a!} f(x)} \; ,$$

where $a! \in O$. Now, let us consider the following processes:

$$\begin{tikzcd} p \arrow[loop left, "b!"] \end{tikzcd} \qquad\qquad \begin{tikzcd} q \arrow[loop left, "b!"] \arrow[loop right, "a!"] \end{tikzcd}$$

where $b! \in O$. Again, $p \mathrel{\mathsf{ioco\underline{s}}} q$ but $f(p) \mathrel{\not\mathsf{ioco\underline{s}}} f(q)$, because $f(p) \xrightarrow{a!} f(p)$ but $f(q) \not\xrightarrow{a!}$.

Example 11. This example indicates that the use of input actions in positive premises of output-emitting rules would invalidate Theorem 4. Let f be defined by the following rules:

$$\frac{x \not\xrightarrow{b?}}{f(x) \xrightarrow{\delta!} f(x)} \qquad\qquad \frac{x \xrightarrow{b?} y}{f(x) \xrightarrow{a!} f(x)} \; ,$$

where $a! \in O$ and $b? \in I$. Now, let us consider the following processes:

$$\begin{tikzcd} p \arrow[loop left, "\delta!"] \arrow[loop right, "b?"] \end{tikzcd} \qquad\qquad \begin{tikzcd} q \arrow[loop right, "\delta!"] \end{tikzcd}$$

Again, $p \mathrel{\mathsf{ioco\underline{s}}} q$ but $f(p) \mathrel{\not\mathsf{ioco\underline{s}}} f(q)$, because $f(p) \xrightarrow{a!} f(p)$ but $f(q) \not\xrightarrow{a!}$.

E Proof of Theorem 5

We start by establishing the following lemma, which states the correctness of the definition of contradictory sets of formulae in Definition 9.

Lemma 4. *Assume that H_1 and H_2 are contradictory sets of transition formulae whose left-hand sides are over distinct variables $x_1, \ldots, x_n$. Let σ be a substitution mapping variables to states in an LTS that satisfies H_1. Then there is no substitution ρ such that $\rho(x_i) = \sigma(x_i)$ for $1 \leq i \leq n$ and ρ satisfies H_2.*

Proof. Since H_1 and H_2 are contradictory, there are $H_1' \subseteq H_1$ and $H_2' \subseteq H_2$ such that H_1' and H_2' contradict each other. We proceed by a case analysis on the possible form H_1' and H_2' may take, following Definition 9.

– Assume that $H_1' = \{x \xrightarrow{a} y\}$ and $H_2' = \{x \overset{a}{\not\longrightarrow}\}$ for some $a \in L$. Since $\sigma(x) \xrightarrow{a} \sigma(y)$, there is no substitution ρ such that $\rho(x) = \sigma(x) \overset{a}{\not\longrightarrow}$.
– Assume that $H_1' = \{x \xrightarrow{b!} y\}$ and $H_2' = \{x \xrightarrow{\delta!} z\}$ for some $b! \in O \setminus \{\delta!\}$. Since $\sigma(x) \xrightarrow{b!} \sigma(y)$ and no state that can perform such a transition is quiescent, there is no substitution ρ such that $\rho(x) = \sigma(x) \xrightarrow{\delta!}$.
– Assume that $H_1' \cup H_2' = \{x \overset{b!}{\not\longrightarrow} \mid b! \in O\}$ for some variable x. Since $\sigma(x)$ satisfies H_1' and every state in an LTS performs at least one output-labelled transition, we have that $\sigma(x)$ cannot satisfy H_2' and we are done.

The cases resulting by swapping the role of H_1' and H_2' in each item above are handled by symmetric arguments. □

We are now ready to prove prove Theorem 5. We will show the two implications separately.

$\Rightarrow$: Assume that $f(\boldsymbol{p}) \xrightarrow{\delta!} p'$, where $\boldsymbol{p} = p_1, \ldots, p_n$ is a sequence of states in some LTS. Then there are a rule

$$\text{R}\ \frac{H}{f(\boldsymbol{x}) \xrightarrow{\delta!} t}$$

and a substitution σ such that $\sigma(f(\boldsymbol{x})) = f(\boldsymbol{p})$, $\sigma(t) = p'$ and σ satisfies the premises in H. By condition $[\delta_1]$ in Definition 10, we have that $t = f(\boldsymbol{y})$ for some vector of variables $\boldsymbol{y}$ such that each y_i is either x_i or $x_i \xrightarrow{\delta!} y_i \in H$. By the requirement on $\xrightarrow{\delta!}$ in Definition 1 and the fact that σ satisfies the premises in H, we infer that $\sigma(y_i) = p_i$ for each $x_i \xrightarrow{\delta!} y_i \in H$. Thus $p' = \sigma(t) = \sigma(f(\boldsymbol{y})) = f(\boldsymbol{p})$. We are therefore left to show that $f(\boldsymbol{p}) \overset{b!}{\not\longrightarrow}$ for each $b! \in O \setminus \{\delta!\}$. To this end, it is enough to prove that if

$$\text{R'}\ \frac{H'}{f(\boldsymbol{x}) \xrightarrow{b!} t'}$$

is a rule for f, then no substitution ρ that agrees with σ over $\boldsymbol{x}$ can satisfy H'. In order to see this, observe that, by condition $[\delta_1]$ in Definition 10, the sets H

and H' are contradictory. Since σ satisfies H, by Lemma 4, no substitution ρ that agrees with σ over $\boldsymbol{x}$ can satisfy H'.

$\Leftarrow$: Assume now that $f(\boldsymbol{p})\overset{b!}{\not\longrightarrow}$ for all $b! \in O \setminus \{\delta!\}$. We will argue that $f(\boldsymbol{p}) \xrightarrow{\delta!} f(\boldsymbol{p})$. Let $\{r_1, \ldots r_n\}$ be the set of output-emitting rules for f not having $\delta!$ as label of their conclusions. Because of our assumption above, for each r_i there is some premise l_i of r_i for some variable x_j in $\boldsymbol{x}$ that p_j does not satisfy. For each $1 \leq i \leq n$, we define $\bar{l}_i$ as follows:

$$\bar{l}_i = \begin{cases} x_j \xrightarrow{a} y_i & \text{if } l_i = x_j \overset{a}{\not\longrightarrow} \\ x_j \overset{a}{\not\longrightarrow} & \text{if } l_i = x_j \xrightarrow{a} y, \text{ for some } y. \end{cases}$$

Here the variables y_i's are all different and distinct from the variables in $\boldsymbol{x}$. By condition $[\delta_2]$ in Definition 10, we have that the set of rules for f includes the rule

$$\text{R}\ \frac{\{\bar{l}_1, \ldots \bar{l}_n\}}{f(\boldsymbol{x}) \xrightarrow{\delta!} f(\boldsymbol{x})} \ .$$

Indeed, no two sets of formulae included in $\{\bar{l}_1, \ldots \bar{l}_n\}$ contradict each other because of the way that set of formulae is constructed.

It is now easy to see that there exists a substitution ρ such that $\rho(f(\boldsymbol{x})) = f(\boldsymbol{p})$ and ρ satisfies $\{\bar{l}_1, \ldots \bar{l}_n\}$. Hence, we conclude that $f(\boldsymbol{p}) \xrightarrow{\delta!} f(\boldsymbol{p})$, and we are done. $\square$