Logical characterisations, rule formats and compositionality for input-output conformance simulation[☆]

Luca Aceto[a,b], Ignacio Fábregas[c], Carlos Gregorio-Rodríguez[d], Anna Ingólfsdóttir[b]

[a] *Gran Sasso Science Institute, L'Aquila, Italy*
[b] *ICE-TCS, School of Computer Science, Reykjavik University, Iceland*
[c] *IMDEA Software Institute, Spain*
[d] *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 Tretmans' 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, rule formats for it 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 Beohar and Mousavi. It also offers a characteristic-formula construction for iocos over finite processes in an extension of the proposed modal logics with greatest fixed points. 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. The general modal decomposition methodology of Fokkink and van Glabbeek is used to show how to check the satisfaction of properties expressed in the logic for iocos in a compositional way for operations specified by rules in the precongruence rule format for iocos.

Keywords: Input-output conformance simulation, modal logic, rule formats,

[☆]Research partially supported by the projects Nominal SOS (project nr. 141558-051) and Open Problems in the Equational Logic of Processes (project nr. 196050-051) of the Icelandic Research Fund; the Spanish projects DArDOS (TIN2015-65845-C3-1-R), TRACES (TIN2015-67522-C3-3-R) and Comunidad de Madrid FORTE-CM (P2018/TCS-4314); the project *MATHADOR* (COGS 724.464) of the European Research Council, and the Spanish addition to *MATHADOR* (TIN2016-81699-ERC).

Email addresses: luca@ru.is,luca.aceto@gssi.it (Luca Aceto), ignacio.fabregas@imdea.org (Ignacio Fábregas), cgr@sip.ucm.es (Carlos Gregorio-Rodríguez), annai@ru.is (Anna Ingólfsdóttir)

compositionality, modal decomposition

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 as 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, [42] and the references therein. Readers interested in an older and very influential strand of MBT research based on finite-state machines can find a wealth of information in the excellent survey paper [32].)

An alternative conformance relation that can be used to underlie MBT is *input-output conformance simulation* (iocos). This relation follows many of the ideas in the definition of ioco. However, iocos is a branching-time semantics based on simulation, whereas ioco is a trace-based semantics. iocos has been introduced, motivated and proved to be an adequate conformance relation for MBT in [20, 21, 22].

Since iocos 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 iocos and its compositionality. More precisely, in Section 3, we provide modal characterisations of iocos in the style of Hennessy and Milner [25]. We offer two modal chacterisations of iocos, which are based on the use of either a 'non-forcing diamond modality' (Theorem 2) or of a 'forcing box modality' (Theorem 4), and compare them with existing logical characterisations for various semantics (Section 3.1) and a logic for ioco proposed by Beohar and Mousavi in [7] (Section 6.1). We also provide a characteristic formula construction for iocos (Proposition 15 in Section 5) for which we need an extension of the logic for iocos with fixed-points (Section 4), and show, by means of an example, that, contrary to what is claimed in [33, Theorem 2], ioco and iocos do *not* coincide even when implementations are input enabled (Section 6).

As argued in [6, 43] 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 ioco$_s$, which is based on the theory of rule formats for structural operational semantics [3]. In Section 7.1, we present a congruence rule format for ioco$_s$ based on the GSOS format proposed by Bloom, Istrail and Meyer [11] (Theorem 21). Some of the conditions of the rule format for ioco$_s$ (Definition 16) have similarities with those of the rule format for XY-simulation given by Beohar and Mousavi in [8]. However, our rule format for ioco$_s$ includes a rather involved 'global' condition that stems from the fact that a specification need only simulate input transitions from an implementation that are labelled with actions that the specification affords. In Section 7.1.1, we present some examples showing that the restrictions of the rule format from Definition 16 cannot be relaxed easily.

A bridge between modal characterisations of process semantics and rule formats for operational semantics is provided by the so-called *modal decomposition* method of [17], whose roots can be traced back to the work by Larsen and Liu [31]. Intuitively, this method allows one to determine whether some process of the form $f(p_1, \ldots, p_n)$ satisfies a formula φ by constructing, from φ and the rules defining the operation f, a collection of properties $\varphi_1, \ldots, \varphi_n$ such that $f(p_1, \ldots, p_n) \models \varphi$ if, and only if, $p_i \models \varphi_i$ for $i \in \{1, \ldots, n\}$. This essentially amounts to providing a compositional model-checking procedure and proof system for the studied process logic with respect to operations specified by rules in a given format.

In Section 7.2 we follow this approach for one of the characterising logics for ioco$_s$ in order to obtain a compositional proof system to decide whether a term built using operations specified by rules in the ioco$_s$ rule format satisfies a formula (Theorem 22). The conditions of the ioco$_s$ rule format play a crucial role in the proof of correctness of the given decomposition method.

Since operations preserving ioco$_s$ need to take quiescence into account properly, we also propose a rule format guaranteeing that operations preserve coherent quiescent behaviour (Theorem 23 in Section 7.3), and show that it is not easy to combine the rule formats for congruence and quiescence (Proposition 24). Finally, Section 8 concludes the paper and presents avenues for future research.

Some of the results in Sections 3, 6 and 7 were presented in the conference paper [2]. In this extended work we offer proofs (some in the appendices) of the results that were announced without proof in [2] and complement the study in those sections with new results, examples and counterexamples. More in detail, the present work contains the following new material.

- In Section 3.1, we provide a comparison between the logics characterising ioco$_s$ and some well-known process logics.
- In Section 4, we develop an extension with fixed-points of the logics for ioco$_s$.
- In Section 5, we define the characteristic formulae for ioco$_s$-processes.
- In Section 7.1.1, we show by means of examples that the conditions for the congruence rule format cannot be relaxed easily.

- In Section 7.2, we apply the modal-decomposition method for ioco$\underline{s}$.
- In Proposition 24, we show that is not easy to combine the congruence and quiescence rule formats by considering the example of the binary merge operator.

2. Preliminaries

The input-output conformance simulation preorder presented in [20, 21, 23] (henceforth referred to as ioco$\underline{s}$) is a semantic relation developed under the assumption that systems have two kinds of actions: 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, $a, b, c \in L$.

A state with no output actions cannot proceed autonomously; such a state is called *quiescent*. Following Tretmans (see, for instance, [39, 42]), 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* $p \xrightarrow{a} q$ *instead of* $(p, a, q) \in \rightarrow$ *and* $p \xrightarrow{a}$*, for* $a \in L$*, if there exists some* $q \in S$ *such that* $p \xrightarrow{a} q$*. Analogously, we will write* $p \not\xrightarrow{a}$*, for* $a \in L$*, if there is no* q *such that* $p \xrightarrow{a} q$*.*

  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* $p \not\xrightarrow{a!}$ *for each* $a! \in O \setminus \{\delta!\}$*.*

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

Contrary to the classic ioco testing theory, in the theory of ioco$\underline{s}$ presented in [20, 21, 22, 23], all actions are assumed to be observable. In this paper, we follow those references and consider only concrete actions.

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 specific 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 following functions over states of an LTS will be used in the remainder of the paper:

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

The definition of input-output conformance simulation given below stems from [20, 21, 23], to which we refer the interested reader for motivation and discussion.

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

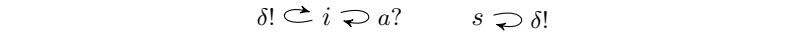
1. $\mathsf{ins}(q) \subseteq \mathsf{ins}(p)$.
2. *For all $a? \in \mathsf{ins}(q)$ and $p' \in S$, if $p \xrightarrow{a?} p'$ then there exists some q' such that $q \xrightarrow{a?} q'$ and $(p', q') \in R$.*
3. *For all $a! \in O$ and $p' \in S$, if $p \xrightarrow{a!} p'$ then there exists some q'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

We define the input-output conformance simulation *(*iocos*) as the largest* iocos*-relation, that is, the union of all* iocos*-relations:*

$$\mathsf{iocos} = \bigcup \{R \mid R \subseteq S \times S,\ R \text{ is an } \mathsf{iocos}\text{-relation}\}.$$

We write $p\ \mathsf{iocos}\ q$ instead of $(p, q) \in \mathsf{iocos}$.

Example 1. *Consider the following processes:*



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

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.*

Throughout the paper we make extensive use of modal logics. A logic over processes is defined by a language to express the formulae in the logic 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 [25].

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 **tt** *and* **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 for all p', $p \xrightarrow{a} p'$ implies $p' \models \varphi$.*

As usual, we extend both boolean connectives $\wedge$ and $\vee$ to finite sets: given a finite index set I and formulae φ_i ($i \in I$), we define $\bigwedge_{i \in I} \varphi_i$ (respectively, $\bigvee_{i \in I} \varphi_i$) as the finite conjunction (respectively, finite disjunction) of the formulae φ_i. We follow the standard convention of considering $\bigwedge_{i \in \emptyset} \varphi_i$ equivalent to **tt** *and $\bigvee_{i \in \emptyset} \varphi_i$ equivalent to* **ff**.

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*

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

Proposition 1. *For each logic $\mathcal{L}$, the binary relation $\leq_{\mathcal{L}}$ is a preorder.*

3. Logic for iocos

In this section we present logics that characterise the iocos relations, both the preorder and its induced equivalence. These logics are subsets of Hennessy-Milner Logic (HML) and are convenient to characterise clearly the discriminating power of the iocos relation. We will use these logics to compare iocos with other behavioural simulation-based relations in the literature in Sections 3.1 and 6. In Section 4, we will address the study of more expressive logics for iocos that are more suitable for the description of system properties.

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

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

where $a? \in I$ and $a! \in O$. The semantics of the constants **tt** *and* **ff**, *of the boolean connectives $\wedge$ (conjunction) and $\vee$ (disjunction), and of the modality $\langle a! \rangle$ (diamond) are defined as usual. The satisfaction relation for the modality $\langle\!|a?|\!\rangle$ is given below:*

$$p \models \langle\!|a?|\!\rangle \phi \textit{ iff } p \xrightarrow{a?}\!\!\!\!\!\!\not\quad \textit{ or } p' \models \phi \textit{ for some } p \xrightarrow{a?} p'.$$

The new modal operator $\langle\!|a?|\!\rangle$ can be read as a *non-forcing* diamond modality: if the input action labelling 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 iocos s to hold, s need only match the input transitions of i that are labelled with input actions that s affords.

Remark 2. *Note that the formula $\langle\!|a?|\!\rangle\mathbf{ff}$ is logically equivalent to the HML formula $[\,a?\,]\mathbf{ff}$. In other words, $p \models \langle\!|a?|\!\rangle\mathbf{ff}$ iff $p \xrightarrow{a?}\!\!\!\!\!\!/\;\;$.*

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

Theorem 2. *For all states i, s in some LTS,*

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

Proof. We prove the two implications separately.

- *Only if implication*: Assume that i iocos s and $i \models \phi$ with $\phi \in \mathcal{L}_{\mathsf{iocos}}$. 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 $s \xrightarrow{a?}\!\!\!\!\!\!/\;\;$, in what follows we may assume that $a? \in \mathsf{ins}(s)$. Since $i\,\mathsf{iocos}\,s$, 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' iocos 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 $R = \{(i, s) \mid i \leq_{\mathcal{L}_{\mathsf{iocos}}} s\}$. We claim that R is an iocos-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 $\mathcal{L}_{\mathsf{iocos}}$ such that $i \models \varphi$ and $s \not\models \varphi$, contradicting the assumption that $i \leq_{\mathcal{L}_{\mathsf{iocos}}} s$. 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, using Remark 2, $i \models \langle\!|a?|\!\rangle\mathbf{ff}$ 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 $a! \in O$ and i' such that $i \xrightarrow{a!} i'$ and $(i', s') \notin R$ for all $s \xrightarrow{a!} s'$. As above, for each $s \xrightarrow{a!} s'$ there exists some formula $\varphi_{s'}$ such that $i' \models \varphi_{s'}$ and $s' \not\models \varphi_{s'}$. In this case, the formula $\varphi = \langle a! \rangle \bigwedge_{s \xrightarrow{a!} s'} \varphi_{s'}$ is such that $i \models \varphi$ but $s \not\models \varphi$. □

Remark 3. *An easy consequence of the above logical-characterisation theorem is that there is no formula in $\mathcal{L}_{\mathsf{iocos}}$ that is logically equivalent to the HML formula $\langle a? \rangle \mathbf{tt}$. Indeed, that formula is satisfied by state i, but not by state s, in Example 1.*

The iocos relation is a preorder and it induces the equivalence relation, $\mathsf{iocos}_{\equiv}$. We can logically characterise this equivalence with the following result.

Corollary 3. *For all states p, q in some LTS,*

$$p \; \mathsf{iocos}_{\equiv} \; q \quad \textit{iff} \quad (\forall \phi \in \mathcal{L}_{\mathsf{iocos}} \quad p \models \phi \textit{ iff } q \models \phi).$$

The logic for iocos we have presented in Definition 5 follows a standard approach to the logical characterisation of simulation semantics; see, for instance, [15, 18]. However, the iocos 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 $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ is defined by the following BNF grammar:*

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

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

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

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 $\langle a? \rangle \mathbf{tt} \wedge [\,a?\,]\phi$.

Remark 4. *Notice that the formula $[\![a?]\!]\mathbf{tt}$ is logically equivalent to $\langle a? \rangle \mathbf{tt}$. In other words, $p \models [\![a?]\!]\mathbf{tt}$ iff $p \xrightarrow{a?}$.*

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

We note that the logics $\mathcal{L}_{\mathsf{iocos}}$ and $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ 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 are at the heart of the proof of the following result, as we will show below.

Theorem 4. *For all states i, s in some LTS, $i \,\mathsf{iocos}\, s$ iff $s \leq_{\widetilde{\mathcal{L}}_{\mathsf{iocos}}} i$.*

Definition 7. *We define the bijection $\mathcal{T} : \mathcal{L}_{\mathsf{iocos}} \to \widetilde{\mathcal{L}}_{\mathsf{iocos}}$ by induction on the structure of formulae in the following way:*

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

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

In order to prove Theorem 4, 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 5. *For each state p in some LTS, and formula $\phi \in \mathcal{L}_{\mathsf{iocos}}$, 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 $\phi \in \mathcal{L}_{\mathsf{iocos}}$. We limit ourselves to giving the proof for the case $\phi = \langle\!|a?|\!\rangle\varphi$. By definition $\mathcal{T}(\langle\!|a?|\!\rangle\varphi) = [\![a?]\!]\mathcal{T}(\varphi)$.

(i): Consider some process p such that $p \models \langle\!|a?|\!\rangle\varphi$. Since $p \models \langle\!|a?|\!\rangle\varphi$ we have that either $p \xrightarrow{a?}\!\!\!\!\!\!\not\;\;$, or there exists $p \xrightarrow{a?} p'$ such that $p' \models \varphi$. If $p \xrightarrow{a?}\!\!\!\!\!\!\not\;\;$, 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): Consider, on the other hand, 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. □

The following corollary will be useful in the proof of Theorem 4.

Corollary 6. *For every state p in some LTS, and all formulae $\phi \in \mathcal{L}_{\mathsf{iocos}}$ and $\psi \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$, the following properties hold:*

(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 the ingredients we need in order to prove Theorem 4, which is the equivalent result to Theorem 2 for $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$.

Proof of Theorem 4. Assume that $i \,\mathsf{iocos}\, s$ and let us consider $\phi \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$ such that $s \models \phi$. By Corollary 6.(ii), $s \not\models \mathcal{T}^{-1}(\phi)$, with $\mathcal{T}^{-1}(\phi) \in \mathcal{L}_{\mathsf{iocos}}$. Now, by Theorem 2, this implies $i \not\models \mathcal{T}^{-1}(\phi)$ and by Lemma 5.ii, $i \models \mathcal{T}(\mathcal{T}^{-1}(\phi)) = \phi$.

On the other hand, let us suppose that for all $\phi \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$, if $s \models \phi$ then $i \models \phi$. By Theorem 2 to show that $i \,\mathsf{iocos}\, s$ it suffices to prove that $i \models \varphi$ implies $s \models \varphi$ for each $\varphi \in \mathcal{L}_{\mathsf{iocos}}$. To this end, let us suppose that $i \models \varphi$. By Corollary 6.(i), we have that $i \not\models \mathcal{T}(\varphi)$. Now, by hypothesis, since $\mathcal{T}(\varphi) \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$ it must also be the case that $s \not\models \mathcal{T}(\varphi)$. Applying again Corollary 6.(i) we obtain that $s \models \varphi$. Thus, we conclude that $i \,\mathsf{iocos}\, s$, finishing the proof. □

As we did in Corollary 3 for $\mathcal{L}_{\mathsf{iocos}}$, we can also easily characterise the equivalence relation $\mathsf{iocos}_{\equiv}$ by using $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$.

Corollary 7. *For all states p, q in some LTS,*

$$p \,\mathsf{iocos}_{\equiv}\, q \quad \textit{iff} \quad (\forall \phi \in \widetilde{\mathcal{L}}_{\mathsf{iocos}} \quad p \models \phi \textit{ iff } q \models \phi).$$

From Corollaries 3 and 7 we can also tell that $\mathsf{iocos}_{\equiv}$ can be logically characterised by means of boolean combinations of the formulae in the logics $\mathcal{L}_{\mathsf{iocos}}$ and $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$. This means that we could write formulae that are in $\mathcal{L}_{\mathsf{iocos}}$ or in $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ and make conjunctions and disjunctions between them, but not nest in the same formula modal operators from $\mathcal{L}_{\mathsf{iocos}}$ and $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$. That is, $\langle\!\langle a? \rangle\!\rangle \mathbf{tt} \wedge [\, x! \,] \mathbf{ff}$ would be acceptable, but $\langle\!\langle a? \rangle\!\rangle [\, x! \,] \mathbf{ff}$, would not. This is reminiscent of the case of mutual simulation and its logical characterisation as opposed to bisimulation equivalence and its logical characterisation [15, 18].

Definition 8. *The logic $\mathcal{L}^{\equiv}_{\mathsf{iocos}}$ is defined by the following BNF grammar:*

$$\phi ::= \psi_{\mathcal{L}_{\mathsf{iocos}}} \mid \psi_{\widetilde{\mathcal{L}}_{\mathsf{iocos}}} \mid \phi \vee \phi \mid \phi \wedge \phi,$$

where $\psi_{\mathcal{L}_{\mathsf{iocos}}} \in \mathcal{L}_{\mathsf{iocos}}$ and $\psi_{\widetilde{\mathcal{L}}_{\mathsf{iocos}}} \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$.

Corollary 8. *For all states p, q in some LTS,*

$$p \,\mathsf{iocos}_{\equiv}\, q \textit{ iff } (\forall \phi \in \mathcal{L}^{\equiv}_{\mathsf{iocos}} \quad p \models \phi \textit{ iff } q \models \phi).$$

3.1. Related logics

Logics provide a systematic way to compare behavioural relations. In this section we relate the characterisations for iocos presented in Section 3 with other logics for coinductively defined relations based on similarity, previously defined in the literature.

First we start with HML (Definition 3), characterising bisimulation equivalence [25], that determines a clear upper bound on the expressiveness of the logics for iocos. In what follows, (strict) inclusions and equalities between logics are up to logical equivalence. For example, $\mathcal{L} \subseteq \mathcal{L}'$ means that, for each $\varphi \in \mathcal{L}$, there is some $\varphi' \in \mathcal{L}'$ that is logically equivalent to φ.

Proposition 9. *$\mathcal{L}_{\mathsf{iocos}} \subset$ HML and $\widetilde{\mathcal{L}}_{\mathsf{iocos}} \subset$ HML.*

Proof. The only non-standard operator used in $\mathcal{L}_{\mathsf{iocos}}$ is the $\langle\!|\, \cdot \,|\!\rangle$ operator ($[\![\cdot]\!]$ in $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$, respectively), but, as observed earlier, formulae $\langle\!|a?|\!\rangle\varphi$ ($[\![a?]\!]\varphi$, respectively) can be expressed in HML as $\langle a?\rangle\varphi \vee [\,a?\,]\mathbf{ff}$ ($\langle a?\rangle\mathbf{tt} \wedge [\,a?\,]\varphi$, respectively). We assume no distinction between input and output actions in HML. As noted before in Remark 3, the HML formula $\langle a?\rangle\mathbf{tt}$ cannot be expressed in $\mathcal{L}_{\mathsf{iocos}}$ and thus $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ cannot express its dual $[\,a?\,]\mathbf{ff}$. □

The simulation and ready simulation preorders [11] have been logically characterised in the literature using subsets of HML, see for instance [12, 15, 18].

Definition 9. *The logics for plain simulation and ready simulation, denoted by $\mathcal{L}_s$ and $\mathcal{L}_{rs}$ respectively, are defined by the following BNF grammars, where $a \in L$:*

$$\begin{array}{lcl} \mathcal{L}_s & : & \phi ::= \mathbf{ff} \mid \mathbf{tt} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a\rangle\phi. \\ \mathcal{L}_{rs} & : & \phi ::= \mathbf{ff} \mid \mathbf{tt} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a\rangle\phi \mid [\,a\,]\mathbf{ff}. \end{array}$$

Remark 5. *In the classic literature, see for instance [11, 12, 18], the logics characterising simulation and ready simulation include neither the disjunction operator $\vee$ nor the constant $\mathbf{ff}$; however, as was already proved in [15], those constructs can be safely added to those logics without altering their discriminating power.*

Proposition 10. *For the logics $\mathcal{L}_s, \mathcal{L}_{rs}$ and $\mathcal{L}_{\mathsf{iocos}}$ the following properties hold:*

1. *$\mathcal{L}_{rs} \supseteq \mathcal{L}_{\mathsf{iocos}}$,*
2. *$\mathcal{L}_{rs} \not\subseteq \mathcal{L}_{\mathsf{iocos}}$,*
3. *$\mathcal{L}_s \not\supseteq \mathcal{L}_{\mathsf{iocos}}$ and*
4. *$\mathcal{L}_s \not\subseteq \mathcal{L}_{\mathsf{iocos}}$.*

Proof. Let us consider a state p with transitions $p \xrightarrow{a?} p$, $p \xrightarrow{b?} p$ and $p \xrightarrow{\delta!} p$, and state q with the transitions $q \xrightarrow{a?} q$ and $q \xrightarrow{\delta!} q$. Observe that $p\,\mathsf{iocos}\,q$ and therefore $\mathcal{L}_{\mathsf{iocos}}(p) \subseteq \mathcal{L}_{\mathsf{iocos}}(q)$ (Theorem 2). This will be used in the proofs of statements 2–4 below. We consider each statement separately.

1. $\mathcal{L}_{rs} \supseteq \mathcal{L}_{\mathsf{iocos}}$. All formulas in $\mathcal{L}_{\mathsf{iocos}}$ are also in $\mathcal{L}_{rs}$.

2. $\mathcal{L}_{rs} \not\subseteq \mathcal{L}_{\mathsf{iocos}}$. Considering states p and q above, we have that $p \,\mathsf{iocos}\, q$ and therefore $p \leq_{\mathcal{L}_{\mathsf{iocos}}} q$. However, the formula $\langle b? \rangle \mathbf{tt}$ in $\mathcal{L}_{rs}$ is satisfied by p and not q. So there is no $\mathcal{L}_{\mathsf{iocos}}$ formula equivalent to it.

3. $\mathcal{L}_{s} \not\supseteq \mathcal{L}_{\mathsf{iocos}}$. Considering states p and q above, we have that q is simulated by p, that is $q \leq_{\mathcal{L}_s} p$, but formula $\langle\!\langle b? \rangle\!\rangle \mathbf{ff}$ is satisfied by q and not by p and therefore $q \not\leq_{\mathcal{L}_{\mathsf{iocos}}} p$. It follows that there is no formula in $\mathcal{L}_s$ that is logically equivalent to $\langle\!\langle b? \rangle\!\rangle \mathbf{ff}$.

4. $\mathcal{L}_{s} \not\subseteq \mathcal{L}_{\mathsf{iocos}}$. Considering again states p and q. We have that $p \,\mathsf{iocos}\, q$ and therefore $p \leq_{\mathcal{L}_{\mathsf{iocos}}} q$, but $p \not\leq_{\mathcal{L}_s} q$ because p satisfies $\langle a? \rangle \mathbf{tt}$ but q does not. This means that $\langle a? \rangle \mathbf{tt}$ cannot be expressed in $\mathcal{L}_{\mathsf{iocos}}$ up to logical equivalence. □

As we already discussed in Remark 5, disjunction does not add any distinguishing power to $\mathcal{L}_{rs}$. However, disjunction is needed for the validity of statement 1 in Proposition 10, as the next lemma formalises.

Lemma 11. *Let $\mathcal{L}'_{rs}$ be defined as*

$$\mathcal{L}'_{rs} : \phi ::= \mathbf{tt} \mid \phi \wedge \phi \mid \langle a \rangle \phi \mid [\, a \,] \mathbf{ff},$$

where $a \in L$. Then, there is no formula in $\mathcal{L}'_{rs}$ such that is logically equivalent to the formula $\langle\!\langle a? \rangle\!\rangle \langle b! \rangle \mathbf{tt} \in \mathcal{L}_{\mathsf{iocos}}$.

Proof. Let ϕ be a formula in $\mathcal{L}'_{rs}$. We will show that ϕ is not equivalent to $\psi = \langle\!\langle a? \rangle\!\rangle \langle b! \rangle \mathbf{tt}$. To this end, note first of all that, up to logical equivalence, ϕ can be written as $\phi = \bigwedge_{i \in I} \langle a_i \rangle \phi_i \wedge \bigwedge_{j \in J} [\, b_j \,] \mathbf{ff}$, for finite sets I and J, actions a_i and b_j and formulae ϕ_i. Notice that if $I = J = \emptyset$, then ϕ is a tautology which is not logically equivalent to $\psi = \langle\!\langle a? \rangle\!\rangle \langle b! \rangle \mathbf{tt}$. Let us assume then that $I \neq \emptyset$ or $J \neq \emptyset$. We distinguish two cases.

First, assume that some b_j is different from $a?$. Let us consider the process p such that $p \xrightarrow{b_j} p$ and $p \xrightarrow{\delta!} p$. We have that $p \models \psi$ but $p \not\models \phi$, which implies that ψ and ϕ are not logically equivalent.

Next, assume that all the b_j's are equal to $a?$. There are two subcases.

- $J \neq \emptyset$. In this case, if there exists some $i \in I$ such that $a_i = a?$, ϕ is unsatisfiable, and thus is not equivalent to ψ. Otherwise, let us consider two subcases. If $I = \emptyset$, ϕ is logically equivalent to $[\, a? \,] \mathbf{ff}$ which is not equivalent to ψ. Otherwise, if $I \neq \emptyset$, the process p such that $p \xrightarrow{a?} p$ and $p \xrightarrow{b!} p$, clearly satisfies ψ but $p \not\models \phi$.

- $J = \emptyset$. In this case, $I \neq \emptyset$. There are two cases: either there exists some $i \in I$ such that $a_i = a?$, or there is not. In the former, the process $\mathrm{nil} \xrightarrow{\delta!} \mathrm{nil}$ satisfies ψ but does not satisfy ϕ. In the latter, let us consider a process q such that $q \xrightarrow{a?} q$ and $q \xrightarrow{\delta!} q$. Clearly $q \models \psi$ but $q \not\models \phi$, which implies that those formulae are not equivalent. □

Besides the classic semantic relations (bisimulation, simulation and ready simulation) there is a less standard relation in the literature, the so-called covariant-contravariant simulation, that is quite close to ioco$_{\underline{s}}$ in its formulation. Let us recall here this relation and its characterising logic.

Definition 10 (Covariant-contravariant simulation [16])**.** *Consider an alphabet A, and let $\{A^r, A^l, A^{bi}\}$ be a partition of this alphabet. A binary relation R over states in an LTS is a covariant-contravariant simulation relation if and only if for each $(p, q) \in R$ the following conditions hold:*

- *for all $a \in A^r \cup A^{bi}$ and all $p \xrightarrow{a} p'$ there exists $q \xrightarrow{a} q'$ with $(p', q') \in R$,*
- *for all $b \in A^l \cup A^{bi}$, and all $q \xrightarrow{b} q'$ there exists $p \xrightarrow{b} p'$ with $(p', q') \in R$.*

We will write $p \lesssim_{CC} q$ if there exists a covariant-contravariant simulation R such that $(p, q) \in R$.

Let us note that the case of $A^l = A^{bi} = \emptyset$ and $A^r = L$ would yield plain simulation. In addition, the case $A^l = A^r = \emptyset$ and $A^{bi} = L$ would yield bisimulation. Covariant-contravariant simulation are also called XY-simulation (for instance in [1]). There the sets X and Y might not be disjoint, but the definitions are equivalent.

Definition 11 (Covariant-contravariant logic [16])**.** *The syntax of the logic for covariant-contravariant, denoted by $\mathcal{L}_{CC}$, is defined by the following BNF grammar:.*

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

where $a \in A^r \cup A^{bi}$ and $b \in A^l \cup A^{bi}$.

There are clear similarities between Definition 10 and the definition of ioco$_{\underline{s}}$ in Definition 2. Nevertheless, as it is not difficult to prove from the characterising logics, the relations $\lesssim_{CC}$ and ioco$_{\underline{s}}$ are not related.

Proposition 12. *The covariant-contravariant simulation and* ioco$_{\underline{s}}$ *are not comparable.*

Proof. Given that for the ioco$_{\underline{s}}$ relation the input and output sets are disjoint, we consider $A^{bi} = \emptyset$. Let us consider the following processes:

- $p \xrightarrow{a?} p'$, $p \xrightarrow{\delta!} p$ and $p' \xrightarrow{b!} p'$;
- $q \xrightarrow{a?} q_1$, $q \xrightarrow{\delta!} q$, $q \xrightarrow{a?} q_2$, $q_1 \xrightarrow{b!} q_1$ and $q_2 \xrightarrow{\delta!} q_2$;
- $r \xrightarrow{a?} r$, $r \xrightarrow{b?} r$ and $r \xrightarrow{\delta!} r$; and
- $s \xrightarrow{a?} s$ and $s \xrightarrow{\delta!} s$.

We discuss the different alternatives:

- If $A^r = I$, $A^l = O$, then $\mathcal{L}_{CC}$ can be expressed as

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

which is not comparable with $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$

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

First, let us consider the processes p and q. $p \lesssim_{CC} q$ which implies that $\mathcal{L}_{CC}(p) \subseteq \mathcal{L}_{CC}(q)$, but $p \models [\![a?]\!][\, \delta! \,]\mathbf{ff}$ whereas $q \not\models [\![a?]\!][\, \delta! \,]\mathbf{ff}$. It follows that there is no formula in $\mathcal{L}_{CC}$ that is logically equivalent to $[\![a?]\!][\, \delta! \,]\mathbf{ff}$.

Analogously, consider r and s. Is clear that r iocos s which implies that $\widetilde{\mathcal{L}}_{\mathsf{iocos}}(r) \subseteq \widetilde{\mathcal{L}}_{\mathsf{iocos}}(s)$, but $r \models \langle b? \rangle \mathbf{tt}$ and $s \not\models \langle b? \rangle \mathbf{tt}$, which implies that there is no formula in $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ that is logically equivalent to $\langle b? \rangle \mathbf{tt}$.

- If $A^r = O$, $A^l = I$, then $\mathcal{L}_{CC}$ can be expressed as

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

that is not comparable to $\mathcal{L}_{\mathsf{iocos}}$

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

First, let us consider p and q. Observe that p iocos q, which implies $\mathcal{L}_{\mathsf{iocos}}(p) \subseteq \mathcal{L}_{\mathsf{iocos}}(q)$. Now, $p \models [\, a? \,]\langle b! \rangle \mathbf{tt}$ but $q \not\models [\, a? \,]\langle b! \rangle \mathbf{tt}$. It follows that there is no formula in $\mathcal{L}_{\mathsf{ioco}}$ that is logically equivalent to $[\, a? \,]\langle b! \rangle \mathbf{tt}$.

Analogously, consider r and s. Is clear that $s \lesssim_{CC} r$ which implies that $\mathcal{L}_{CC}(s) \subseteq \mathcal{L}_{CC}(r)$, but $s \models \langle\!| b? |\!\rangle [\, \delta! \,]\mathbf{ff}$ and $r \not\models \langle\!| b? |\!\rangle [\, \delta! \,]\mathbf{ff}$, which implies that there is no formula in $\mathcal{L}_{CC}$ that is logically equivalent to $\langle\!| b? |\!\rangle [\, \delta! \,]\mathbf{ff}$. □

The comparison between the logics characterising the relations iocos and Tretmans' ioco is of particular interest. We will delay this comparison till Section 6, as we will make use of some of the results in the following sections.

4. Adding fixed points to the logics

Even if an LTS is image-finite and has a finite number of states, it can capture infinite behaviours, for example by means of loops. As is well known, the logics presented in Section 3 can only describe properties of finite fragments of process computations and can be made more expressive by extending them with fixed-point operators.

Here we only explicitly show how to add the greatest fixed-point operator to one of the characterising logics for iocos, $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$, presented in Section 3. We will make use of this extended logic in Section 5 to define the characteristic formula for processes with respect to iocos.

In this section we follow the approach of the so-called equational μ-calculus (see, for example, [4, 30] to which we refer the reader for further details). We assume a countably infinite set Var of formula variables. The meaning of formula variables is specified by means of a declaration. A declaration is a function $D : \mathsf{Var} \to \widetilde{\mathcal{L}}_{\mathsf{iocos}}$ that associates a formula $D(X)$ with each variable X. Intuitively, if $D(X) = \phi$, then X stands for the largest solution of the equation $X = \phi$. In what follows, we are only interested in the restriction of a declaration to a finite collection of formula variables, and we express such a declaration as a system of equations $X_1 = \phi_1, \ldots, X_n = \phi_n$, with $n \geq 1$.

We interpret the language over the set S of processes in some LTS. Since a formula φ may contain formula variables, its semantics—that is, the set of processes in S that satisfy φ—is defined relative to an environment $\sigma : \mathsf{Var} \to \mathcal{P}(S)$. Intuitively, σ assigns to each variable the set of processes in S for which the variable holds true.

As is well known, the set of all environments $[\mathsf{Var} \to \mathcal{P}(S)]$ is a complete lattice with respect to the partial order induced by pointwise set inclusion.

Definition 12. *The syntax of the logic for* iocos *with greatest fixed points, denoted by* $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}}$*, is defined by the following BNF grammar*

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

where $a? \in I$*,* $a! \in O$ *and* $X \in \mathsf{Var}$*.*

The semantics of $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}}$ is given by:

- $(\sigma, p) \models X$ iff $p \in \sigma(X)$.
- $(\sigma, p) \models \mathbf{tt}$ for all p.
- $(\sigma, p) \models \varphi_1 \wedge \varphi_2$ iff $(\sigma, p) \models \varphi_1$ and $(\sigma, p) \models \varphi_2$.
- $(\sigma, p) \models \varphi_1 \vee \varphi_2$ iff $(\sigma, p) \models \varphi_1$ or $(\sigma, p) \models \varphi_2$.
- $(\sigma, p) \models [\,a!\,]\phi$ iff $(\sigma, p') \models \phi$ for each $p \xrightarrow{a!} p'$.
- $(\sigma, p) \models [\![a?]\!]\phi$ iff $p \xrightarrow{a?}$ and $(\sigma, p') \models \phi$, for each $p \xrightarrow{a?} p'$.

The semantics of each formula φ is therefore the function $(\!|\varphi|\!) : [\mathsf{Var} \to \mathcal{P}(S)] \to \mathcal{P}(S)$ defined thus:

$$(\!|\varphi|\!)\sigma = \{p \mid (\sigma, p) \models \varphi\}.$$

A declaration function D induces an endofunction $(\!|D|\!)$ over the complete lattice $[\mathsf{Var} \to \mathcal{P}(S)]$ defined by

$$((\!|D|\!)\sigma)(X) = (\!|D(X)|\!)\sigma.$$

Proposition 13. *The function* $(\!|D|\!)$ *is monotone. Indeed, so is* $(\!|\varphi|\!)$ *for each formula* φ*.*

Proof. Since $[\![a?]\!]\varphi$ is logically equivalent to the HML formula $\langle a?\rangle \mathbf{tt} \wedge [\,a?\,]\varphi$, the result follows straightforwardly from the monotonicity of the semantic counterparts of the operators in HML (see [30], for example). □

Since $(\!|D|\!)$ is a monotone endofunction over a complete lattice, it has a greatest (and a least) fixed point $(\!|D|\!)_{\max}$ by the Knaster-Tarski Theorem [40].

Theorem 14. *For all states i, s in some LTS,*

$$i \;\mathsf{iocos}\underline{\mathsf{s}}\; s \quad \textit{iff} \quad \forall \phi \in \widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}} \quad s \models \phi \Rightarrow i \models \phi.$$

Proof. Since $\widetilde{\mathcal{L}}_{\mathsf{iocos}\underline{\mathsf{s}}}$ is a sublogic of $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}}$, the *if* implication of the proof follows by Theorem 4. For the *only if* implication, the proof follows by applying the standard techniques for μ-calculus and bisimulation equivalence (see, for example, [38, Section 5.4]). □

Remark 6. *We showed in Section 3 how $\mathcal{L}_{\mathsf{iocos}\underline{\mathsf{s}}}$ and $\widetilde{\mathcal{L}}_{\mathsf{iocos}\underline{\mathsf{s}}}$ are dual logics (Corollary 6). Analogously to the construction of $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}}$ from $\widetilde{\mathcal{L}}_{\mathsf{iocos}\underline{\mathsf{s}}}$, we could add μ, the least fixed point operator, to the logic $\mathcal{L}_{\mathsf{iocos}\underline{\mathsf{s}}}$, for defining $\mathcal{L}^{\mu}_{\mathsf{iocos}\underline{\mathsf{s}}}$.*

Logics like $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}}$ can be used in model checking in order to evaluate if an implementation satisfies some desired properties. There is already a tool support for $\mathsf{iocos}\underline{\mathsf{s}}$ implemented in the **mCRL2** tool set [14, 22, 24], including a minimisation algorithm. This allows one to model check $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}}$ using the **mCRL2** tool. In fact, the built-in logic used in **mCRL2** is the *first-order modal μ-calculus* [27], which is an extension of the μ-calculus where the actions are allowed to include data parameters. We omit that extension here, but we show some examples of properties that can be expressed with our logic.

Example 2. *Let us illustrate some useful properties that can be expressed using the logic $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}\underline{\mathsf{s}}}$.*

- *In the theory of* ioco*, implementations are supposed to be input enabled, that is, each of their reachable states should be able to perform every input action. This requirement can be expressed as follows:*

$$X = \bigwedge_{a?\in I} [\![a?]\!]X \wedge \bigwedge_{b!\in O} [\,b!\,]X.$$

- *The next formula formalises that a property φ must hold invariantly over input-enabled systems:*

$$X = \varphi \wedge \Big(\bigwedge_{a?\in I} [\![a?]\!]X \wedge \bigwedge_{o!\in O} [\,o!\,]X\Big).$$

  That is, φ must be true, and must remain true after performing any input or output action.

5. Characteristic formulae

Once we have logically characterised iocos, it is natural to search for a single logical formula that characterises completely the behaviour of a process up to that preorder; this is what is called a *characteristic formula* [19, 37]. In this section we follow [4] in order to define such a formula for processes with respect to iocos. We will consider the logic from Section 4 with its greatest-fixed-point interpretation given therein.

First, let us define the notion of characteristic formula for the particular case of the iocos-semantics.

Definition 13. *A formula χ_s is* characteristic *for s (with respect to iocos) iff for all i it holds that $i \models \chi_s$ if, and only if, $i \,\mathsf{iocos}\, s$. (Note that this also implies that $s \models \chi_s$.)*

The next proposition states the explicit definition of characteristic formulae in the iocos framework.

Proposition 15. *The characteristic formula for a process q in a finite LTS can be obtained recursively as:*

$$\chi_q = \bigwedge_{a? \in \mathsf{ins}(q)} [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} \chi_{q'} \wedge \bigwedge_{a! \in O} [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} \chi_{q'}, \tag{1}$$

that is, letting D be the declaration defined by the equations of the form (1), we have that: $((\!|D|\!)_{\max}, p) \models \chi_q$ iff $p \,\mathsf{iocos}\, q$.

Remark 7. *Note that χ_q constrains the behaviour of a process that satisfies it only for those input actions that q can perform, exactly in the same way as the iocos relation does.*

In order to prove Proposition 15, we follow the general theory described in [4]. The concrete goal is to apply Corollary 3.4 in [4]. As a first step, we define iocos as the greatest fixed-point of a monotone function, denoted by $\mathcal{F}_{io}$, over the set of binary relations over S, the set of states of an LTS.

Definition 14. *For each $R \subseteq S \times S$, we have that $(p, q) \in \mathcal{F}_{io}(R)$ iff*

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

Next, we show a result connecting $\mathcal{F}_{io}$ and χ defined in Proposition 15.

Lemma 16. *Let $R \subseteq S \times S$ and $p, q \in S$. Consider the set of variables $\mathsf{Var} = \{X_p \mid p \in S\}$ and let σ_R be the interpretation defined as $\sigma_R(X_q) = \{p \mid (p,q) \in R\}$. Then, $(p,q) \in \mathcal{F}_{io}(R)$ iff*

$$(\sigma_R, p) \models \bigwedge_{a? \in \mathsf{ins}(q)} [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'} \wedge \bigwedge_{a! \in O} [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} X_{q'},$$

Proof. First, suppose that $(p,q) \in \mathcal{F}_{io}(R)$. We will show that

$$(\sigma_R, p) \models \bigwedge_{a? \in \mathsf{ins}(q)} [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'} \wedge \bigwedge_{a! \in O} [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} X_{q'}.$$

To this end, let $a? \in \mathsf{ins}(q)$. We show first that $(\sigma_R, p) \models [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'}$. Indeed, since $(p,q) \in \mathcal{F}_{io}(R)$, $\mathsf{ins}(q) \subseteq \mathsf{ins}(p)$. Hence, $a? \in \mathsf{ins}(p)$ and therefore $p \xrightarrow{a?}$. Assume now that $p \xrightarrow{a?} p'$, for some p', we claim that $(\sigma_R, p') \models \bigvee_{q \xrightarrow{a?} q'} X_{q'}$. Indeed, since $(p,q) \in \mathcal{F}_{io}(R)$, $a? \in \mathsf{ins}(q)$ and $p \xrightarrow{a?} p'$, there exists some q' such that $q \xrightarrow{a?} q'$ with $(p', q') \in R$, that is, such that $p' \in \sigma_R(X_{q'})$. Thus, we have that $(\sigma_R, p) \models [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'}$.

Next, let us see that $(\sigma_R, p) \models \bigwedge_{a! \in O} [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} X_{q'}$. Analogously to the previous case let us assume that $p \xrightarrow{a!} p'$, for some p'. We claim that $(\sigma_R, p') \models \bigvee_{q \xrightarrow{a?} q'} X_{q'}$. Indeed, since $(p,q) \in \mathcal{F}_{io}(R)$ and $p \xrightarrow{a!} p'$, there exists some q' such that $q \xrightarrow{a!} q'$ with $(p', q') \in R$, that is, such that $p' \in \sigma_R(X_{q'})$. Thus, we have that $(\sigma_R, p) \models [\,a!\,] \bigvee_{q \xrightarrow{a?} q'} X_{q'}$. This completes the proof of the first implication of the theorem.

Now, let us suppose that

$$(\sigma_R, p) \models \bigwedge_{a? \in \mathsf{ins}(q)} [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'} \wedge \bigwedge_{a! \in O} [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} X_{q'}.$$

We prove that $(p,q) \in \mathcal{F}_{io}(R)$. First, since for each $a? \in \mathsf{ins}(q)$, $(\sigma_R, p) \models [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'}$, we obtain that $\mathsf{ins}(q) \subseteq \mathsf{ins}(p)$.

Next, let us consider some $a? \in \mathsf{ins}(q)$. Since $(\sigma_R, p) \models [\![a?]\!] \bigvee_{q \xrightarrow{a?} q'} X_{q'}$, we have that $(\sigma_R, p') \models \bigvee_{q \xrightarrow{a?} q'} X_{q'}$ for every $p \xrightarrow{a?} p'$. Hence, for every $p \xrightarrow{a?} p'$, there exists some $q \xrightarrow{a?} q'$ such that $(\sigma_R, p') \models X_{q'}$, that is, for every $p \xrightarrow{a?} p'$, there exists some $q \xrightarrow{a?} q'$ such that $(p', q') \in R$.

Finally, let us consider $a! \in O$, since $(\sigma_R, p) \models [\,a!\,] \bigvee_{q \xrightarrow{a!} q'} X_{q'}$, we also have that $(\sigma_R, p') \models \bigvee_{q \xrightarrow{a!} q'} X_{q'}$ for every $p \xrightarrow{a!} p'$. Hence, for each $p \xrightarrow{a!} p'$ there exists $q \xrightarrow{a!} q'$ such that $(\sigma_R, p') \models X_{q'}$, that is, such that $(p', q') \in R$. This completes the proof of the lemma. □

In the light of Lemma 16, we can apply Corollary 3.4 in [4] to prove Proposition 15, which gives the explicit definition of the characteristic formula for iocos.

6. The relation between iocos and ioco

Input-output conformance (ioco) was introduced by Tretmans in [41]. 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 formalised below in a setting in which all actions are observable.

Definition 15. *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 $\mathsf{ioco} \subseteq S \times S$ is defined as:*

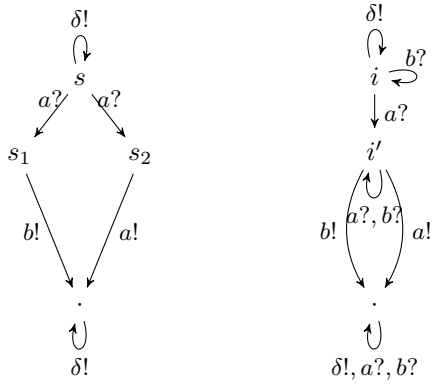
$$i \,\mathsf{ioco}\, s \text{ iff } \mathsf{Out}(i \text{ after } \sigma) \subseteq \mathsf{Out}(s \text{ after } \sigma), \text{ for all } \sigma \in \mathsf{traces}(s).$$

As shown in [20, Theorem 1], iocos is included in ioco.

In the setting of Tretmans' standard ioco theory [41], only input-enabled implementations are considered. We recall that 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 $a? \in I$.

Theorem 2 in [33] states that if i is input enabled, $i \,\mathsf{ioco}\, s$ implies $i \,\mathsf{iocos}\, s$. This means that, when restricted to input-enabled implementations, ioco and iocos coincide, and therefore the logics characterising iocos presented in this paper also characterise ioco over that class of LTSs. Unfortunately, however, Theorem 2 in [33] does *not* hold, as shown in the following example.

Example 3. *Let s and i be defined as follows, where we assume that $I = \{a?, b?\}$.*



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, $i \not\mathsf{iocos}\, s$ because each* iocos *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* iocos*-relation because $i' \xrightarrow{a!}$ and $i' \xrightarrow{b!}$, but $s_1 \not\xrightarrow{a!}$ and $s_2 \not\xrightarrow{b!}$.*

However, as one might expect, Theorem 2 in [33] becomes true when the specification s is deterministic. An LTS is deterministic whenever there is only one possible successor for each action, that is, if for all $p, p', p'' \in S$ and $a \in L$, if $p \xrightarrow{a} p'$ and $p \xrightarrow{a} p''$, then $p' = p''$.

Proposition 17. *Let i be an input-enabled LTS and s be a deterministic LTS. If i ioco s then i iocos s.*

Proof. Assume that i ioco s. We will show that the relation

$$R = \{(p, q) \mid \exists \sigma \in \mathsf{traces}(s) \text{ such that } p \in i \text{ } \mathsf{after} \text{ } \sigma \text{ and } q \in s \text{ } \mathsf{after} \text{ } \sigma\}$$

is a iocos-relation. First, we observe that $(i, s) \in R$ since $\varepsilon \in \mathsf{traces}(s)$. Now, we show that any pair $(p, q) \in R$ satisfies the conditions of Definition 2.

- Trivially, $\mathsf{ins}(q) \subseteq \mathsf{ins}(p)$. Indeed, since i is input-enabled, $\mathsf{ins}(q) \subseteq I = \mathsf{ins}(p)$.
- Let $a? \in \mathsf{ins}(q)$ and assume that $p \xrightarrow{a?} p'$ for some p'. Since $a? \in \mathsf{ins}(q)$, by definition there exists some q' such that $q \xrightarrow{a?} q'$, so $\sigma a? \in \mathsf{traces}(s)$. Hence, $p' \in i \text{ } \mathsf{after} \text{ } \sigma a?$ and $q' \in s \text{ } \mathsf{after} \text{ } \sigma a?$, that is, $(p', q') \in R$.
- Let $b! \in O$ be such that $p \xrightarrow{b!} p'$, that is, $b! \in \mathsf{outs}(p) \subseteq \mathsf{Out}(i \text{ } \mathsf{after} \text{ } \sigma)$. Now, since i ioco s and $\sigma \in \mathsf{traces}(s)$, we have that $\mathsf{Out}(i \text{ } \mathsf{after} \text{ } \sigma) \subseteq \mathsf{Out}(s \text{ } \mathsf{after} \text{ } \sigma)$. Hence, $b! \in \mathsf{Out}(s \text{ } \mathsf{after} \text{ } \sigma)$ and, since s is deterministic, $b! \in \mathsf{outs}(q)$ so there exists some q' such that $q \xrightarrow{b!} q'$. Summing up, we have that $\sigma b! \in \mathsf{traces}(s)$, $p' \in i \text{ } \mathsf{after} \text{ } \sigma b!$ and $q' \in s \text{ } \mathsf{after} \text{ } \sigma b!$, that is, $(p', q') \in R$. □

Remark 8. *The construction of the relation R used in the proof of the above proposition is akin to the definition of 'coinductive* ioco*' employed in the proof of Theorem 3 in [36]. In fact, the use of such relations in the algorithmics of decorated trace semantics, such as* ioco*, as well as failure and testing equivalences, can be traced at least as far back as [13].*

6.1. The relation with a logic for ioco

In [7] Beohar and Mousavi introduced an explicit logical characterisation of ioco. This characterisation uses a non-standard modal operator reminiscent of our $[\![\cdot]\!]$, denoted by $[\!|\cdot|\!]$[1]. However, output actions can also be used as labels of $[\!|\cdot|\!]$. This modality can be extended to traces σ as follows: $p \models [\!|\sigma|\!]\phi$ if, and only if, $p \xrightarrow{\sigma}$ and $p' \models \phi$, for each p' such that $p \xrightarrow{\sigma} p'$. (Note that, for the particular case of input actions $a?$, the semantics of $[\!|a?|\!]$ coincides with that of $[\![a?]\!]$.)

[1] In fact, the symbol used to denote the operator $[\!|\cdot|\!]$ in [7] 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 explicit logical characterisation of ioco given in [7] is defined by means of two different subclasses of logical formulae. The first subclass permits only formulae of the form $[\![\sigma]\!][\,b\,]\mathbf{ff}$, where σ is a trace and b is an output action.

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

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

Lemma 18 ([7])**.** *For each sequence of actions σ, output action b and process p the following statements hold:*

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

The resulting logical characterisation theorem by Beohar and Mousavi for ioco is as follows.

Theorem 19 ([7])**.** *i ioco s iff, for all $\sigma \in L^*$, $b \in O$, if $s \models [\![\sigma]\!][\,b\,]\mathbf{ff}$, then $i \models [\,\sigma\,][\,b\,]\mathbf{ff}$.*

Theorem 4 in this paper is the counterpart of the above result for iocos. Note, however, that Theorem 19 is not a classic modal characterisation result (as it is the case of, for example, Theorem 4) 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.

As we will now argue, the logics for ioco and iocos 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 $[\,\sigma\,][\,b\,]\mathbf{ff}$, with σ a trace, can be expressed in $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ since in an input-enabled scenario $[\![a?]\!]$ has the same semantics as $[\,a?\,]$. On the other hand, it is not possible to define a formula $\phi \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$ that captures Lemma 18(1). Indeed, by way of example, consider $\phi = [\![x!]\!][\,b!\,]\mathbf{ff}$. 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 $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ a formula ψ whose semantics coincides with that of $[\![x!]\!][\,b!\,]\mathbf{ff}$. Let

$$a! \circlearrowright s \circlearrowleft x! \qquad\qquad a! \circlearrowright i$$

It is easy to see that i iocos s, but $s \models \psi$ and $i \not\models \psi$. In other words, ψ is a formula that distinguishes processes related by iocos. Hence, such a formula ψ cannot be expressed in any logic that characterises iocos.

On the other hand, let us consider the two processes of Example 3 and the formula $\phi = [\![a?]\!]([\,a!\,]\mathbf{ff} \vee [\,b!\,]\mathbf{ff}) \in \widetilde{\mathcal{L}}_{\mathsf{iocos}}$. As we already stated in Example 3, i ioco s, but $s \models \phi$ and $i \not\models \phi$. Hence, ϕ can distinguish processes that are ioco-related.

Now, since iocos implies ioco [20], we can use $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}}$ (see Section 4) to express distinguishing formulae whenever two processes are not ioco-related.

Proposition 20. *If $i \not\mathsf{ioco}\, s$, then there exists a formula $\psi \in \widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}}$ such that $s \models \psi$ but $i \not\models \psi$.*

Proof. Straightforward from the fact that $i\ \mathsf{iocos}\ s$ implies $i\ \mathsf{ioco}\ s$, the logical characterisations of $\mathcal{L}_{\mathsf{iocos}}$ in Theorem 2 and $\widetilde{\mathcal{L}}_{\mathsf{iocos}}$ in Theorem 4. □

Remark 9. *As we commented on Remark 6, a dual logic to $\widetilde{\mathcal{L}}^{\nu}_{\mathsf{iocos}}$ is $\mathcal{L}^{\mu}_{\mathsf{iocos}}$. Therefore, analogously, if $i \not\mathsf{ioco}\ s$, then there exists a formula $\phi \in \mathcal{L}^{\mu}_{\mathsf{iocos}}$ such that $i \models \phi$ but $s \not\models \phi$.*

Proposition 20 can be applied, in particular, for χ_s, the characteristic formula of s (Definition 13). Hence, if $i \not\mathsf{ioco}\ s$, we obtain that $s \models \chi_s$ and $i \not\models \chi_s$. That is, the characteristic formula of s can be used as a distinguishing formula. This way we have an alternative, albeit incomplete, criterion for checking if an implementation does not conform to a specification with respect to ioco: (i) first, build χ_s, the characteristic formula for s; (ii) check if $i \models \chi_s$ (iocos satisfaction); (iii) now, if $i \not\models \chi_s$ then, we have that $i \not\mathsf{ioco}\ s$.

7. Rule formats for iocos

In this section we focus on the study of structural properties of iocos, with emphasis on its compositionality and on compositional proof systems for its characterising logic $\mathcal{L}_{\mathsf{iocos}}$. We start, in Section 7.1, by defining a precongruence rule format for iocos. That rule format is shown to ensure compositionality with respect to iocos for a subset of the operators defined using rules in the GSOS format proposed by Bloom, Istrail and Meyer [11]. The rule format provides a sufficient condition for compositionality; we show with counterexamples how the rules we propose cannot be easily relaxed without jeopardizing the compositionality result for iocos.

Next, in Section 7.2, we use the logical characterisation of Section 3 and the general *modal decomposition* methodology of Fokkink and van Glabbeek [17]. For the operators in the rule format of Section 7.1, we show how to check the satisfaction of a logical formula in $\mathcal{L}_{\mathsf{iocos}}$ in a compositional way.

Finally, quiescent behaviour is an important issue in the theory of $\mathsf{ioco}/\mathsf{iocos}$. In Section 7.3 we develop a rule format to ensure coherent quiescent behaviour in the sense of Definition 1.

The restriction in this section to GSOS rules is partly justified by our wish to have a purely syntactic rule format and by the undecidability results presented in [28]. In what follows, we assume that the reader is familiar with the standard notions of signature and terms over a signature.

7.1. Congruence rule format

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:

$$\frac{\{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\} \cup \{x_i \xrightarrow{b_{ik}} \not\rightarrow \mid 1 \leq i \leq n, 1 \leq k \leq \ell_i\}}{f(\vec{x}) \xrightarrow{a} C[\vec{x}, \vec{y}]} \quad (2)$$

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, $C[\vec{x}, \vec{y}]$ 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 in the conclusion* of the rule is $f(\vec{x})$.

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 $\vec{p}$ (with typical entry p_i) and $\vec{q}$ (with entries q_{ij}), there is a transition $f(\vec{p}) \xrightarrow{a} C[\vec{p}, \vec{q}]$ if, and only if, there is an f-defining rule of the form (2) such that

- $p_i \xrightarrow{a_{ij}} q_{ij}$ for each $1 \leq i \leq n$ and $1 \leq j \leq m_i$ and
- $p_i \xrightarrow{b_{ik}} \not\rightarrow$ 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 16. *An operation f in a GSOS language is in* 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 $1 \leq i \leq n$, the positive trigger for variable x_i in r'' is included in the positive trigger for variable x_i in r;*
   (b) *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;*

(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 iocos-format *if so is each of its operations.*

Next, we state the main result of this section.

Theorem 21. iocos *is a precongruence for each GSOS language in* iocos *format.*

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

As an example of application of the above result, we show that the merge operator from [6] can be expressed in our rule format.

Example 4. *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* $1 \leq i \leq n$*. In [6] it 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.*

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

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

It is immediate to check that the above rules are in iocos*-format. Therefore the above theorem yields that the merge operator preserves* iocos.

In the following examples, we discuss whether the rules specifying (variations on) some classic process-algebraic operations and those considered in [43] meet the constraints of our rule format.

Example 5 (Nondeterministic choice)**.** *The following rules describe the behaviour of the nondeterministic choice operation considered in [23, Figure 1], which is a minor variation on the classic CCS choice [35]:*

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 + x_2 \xrightarrow{a} y_1} \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 + x_2 \xrightarrow{a} y_2} \qquad \frac{x_1 \xrightarrow{\delta!} y_1,\ x_2 \xrightarrow{\delta!} y_2}{x_1 + x_2 \xrightarrow{\delta!} y_1 + y_2} ,$$

where $a \in L \setminus \{\delta!\}$. (Note that the third rule above is an equivalent reformulation of rule VII given in [23, Figure 1], which is not in GSOS format.)

These rules are not in ioco$\underline{s}$*-format. To see this, consider the $a?$-emitting rules for some input action $a?$, namely*

$$r = \frac{x_1 \xrightarrow{a?} y_1}{x_1 + x_2 \xrightarrow{a?} y_1} \quad \text{and} \quad r' = \frac{x_2 \xrightarrow{a?} y_2}{x_1 + x_2 \xrightarrow{a?} y_2}.$$

Those rules do not satisfy condition 2 in Definition 16. Indeed, the only possible choice for rule r'' is r', which satisfies neither requirement 2a nor requirement 2c.

It turns out that nondeterministic choice does not preserve ioco$\underline{s}$ *and the reader will find it easy to construct an example witnessing the fact that* ioco$\underline{s}$ *is not a precongruence with respect to $+$ by considering processes that do not satisfy the proviso of [23, Proposition 6.2].*

Example 6 (Interleaving)**.** *The binary version of the merge operator from Example 4 is essentially the synchronous parallel composition of its arguments. Another standard notion of parallel composition operator is one that interleaves the computational steps of its arguments [26]. Ignoring quiescence, the following rules describe the behaviour of the interleaving operator:*

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \mid x_2 \xrightarrow{a} y_1 \mid x_2} \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 \mid x_2 \xrightarrow{a} x_1 \mid y_2}$$

where $a \in L \setminus \{\delta!\}$.

These rules are not in ioco$\underline{s}$*-format. To see this, consider the $a?$-emitting rules for some input action $a?$, namely*

$$r = \frac{x_1 \xrightarrow{a?} y_1}{x_1 \mid x_2 \xrightarrow{a?} y_1 \mid x_2} \quad \text{and} \quad r' = \frac{x_2 \xrightarrow{a?} y_2}{x_1 \mid x_2 \xrightarrow{a?} x_1 \mid y_2}.$$

Those rules do not satisfy condition 2 in Definition 16. Indeed, the only possible choice for rule r'' is r', which satisfies neither requirement 2a nor requirement 2c.

Again, it turns out that the interleaving operation does not preserve ioco$\underline{s}$ *and the reader will find it easy to construct an example witnessing the fact that* ioco$\underline{s}$ *is not a precongruence with respect to $|$. The same applies to the unrestricted version of the parallel composition operator considered in [43, Definition 2.3] in the setting of* ioco*. Note that the notion of parallel composition considered in that reference is a partial operation, as it is only defined for arguments whose sets of input actions and whose sets of output actions are disjoint.*

Example 7 (Relabelling)**.** *In this example, we consider a variation on the CCS relabelling operator [35] that is appropriate in the setting of labelled transition systems with inputs and outputs. A* relabelling *is a function $f : L \to L$ that*

maps input actions to input actions, output actions to output actions and such that $f(\delta!) = \delta!$. The relabelling operator $\_[f]$ associated with a relabelling f is specified using the following rules:

$$\frac{x \xrightarrow{a} y}{x[f] \xrightarrow{f(a)} y[f]} \qquad \text{where } a \in L.$$

If f is injective over the set of input actions I, then the above rules are in iocos *format and Theorem 21 yields that the operator $\_[f]$ preserves* iocos*.*

On the other hand, assume that $f(a?) = f(b?) = a?$ for two distinct input actions $a?$ and $b?$. Then, the collection of rules for $\_[f]$ includes the rules

$$r = \frac{x \xrightarrow{a?} y}{x[f] \xrightarrow{a?} y[f]} \quad \text{and} \quad r' = \frac{x \xrightarrow{b?} y}{x[f] \xrightarrow{a?} y[f]}.$$

Those rules do not satisfy condition 2 in Definition 16. Indeed, choosing r' as r'' violates requirement 2a and choosing r as r'' violates requirement 2c. Therefore the rules for $\_[f]$ are not in iocos*-format. The reader will have no trouble in constructing an example witnessing the fact that, for such a relabelling f, the operator $\_[f]$ does not preserve* iocos*.*

Example 8 (Restriction)**.** *The restriction operator $\_ \setminus A$, where A is included in $L \setminus \{\delta!\}$, is specified using the following rules:*

$$\frac{x \xrightarrow{a} y}{x \setminus A \xrightarrow{a} y \setminus A} \qquad \text{where } a \notin A.$$

It is easy to see that the above rules are in iocos*-format and Theorem 21 yields that the restriction operator $\_ \setminus A$ preserves* iocos*.*

The main lesson one can draw from the examples we have presented is that, similarly to ioco, the iocos relation is not algebraically well behaved. Indeed, a variety of operators only preserve iocos if one makes some assumptions on the sets of input and output actions of their arguments.

The operational specification of the hiding operator **hide** V **in** $\_$, where $V \subseteq O$, presented in [43, Definition 2.3] requires an extension of the theory of iocos with the internal action $\tau \notin L$. We briefly discuss how this extension can be carried out and how to extend the iocos-format to cover the hiding operator in Section 8.

7.1.1. The rule format cannot be relaxed easily

Here we present some examples showing that the restrictions of the rule format from Definition 16 cannot be relaxed easily.

Example 9. *This example indicates that the use of input actions in negative premises of input-emitting rules would invalidate Theorem 21. Let f be defined by the following rules:*

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

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

$$\delta! \circlearrowright i \circlearrowleft a? \qquad s \circlearrowleft \delta!$$

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

Example 10. *This example indicates that the use of output actions in positive premises of input-emitting rules would invalidate Theorem 21. Let f be defined by the following rules:*

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

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

$$c! \circlearrowright p \qquad b! \circlearrowright q \circlearrowleft c!$$

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

Example 11. *This example indicates that not meeting requirement 2a in Definition 16 would invalidate Theorem 21. 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 16. 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 9:

$$\delta! \circlearrowright i \circlearrowleft a? \qquad s \circlearrowleft \delta!$$

We know that i iocos s*. On the other hand, we claim that* $f(i)$ $\mathsf{ioc\acute{o}s}$ $f(s)$*. 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 $\mathsf{ioc\acute{o}s}$ $f(s)$ *(because* $a? \notin \mathsf{ins}(0)$*), this implies that* $f(i)$ $\mathsf{ioc\acute{o}s}$ $f(s)$*.*

Note that requirements 2b and 2c in Definition 16 are instead met by taking $r' = r''$*.*

Example 12. *This example indicates that not meeting requirement 2b in Definition 16 would invalidate Theorem 21. 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 16. 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:

$$c! \circlearrowright p \circlearrowleft a? \qquad\qquad b! \circlearrowright q \circlearrowleft c!$$

where $c! \in O$*. Again,* p iocos q*, but* $f(p)$ $\mathsf{ioc\acute{o}s}$ $f(q)$*. 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 $\mathsf{ioc\acute{o}s}$ $f(q)$*.*

Note that requirements 2a and 2c in Definition 16 are instead met by taking $r' = r''$*.*

Example 13. *This example indicates that not meeting requirement 2c in Definition 16 would invalidate Theorem 21. 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 16. 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:

$$a?,\delta! \circlearrowright p \xrightarrow{b?} 0 \circlearrowright \delta! \qquad\qquad q \circlearrowright a?,\delta!$$

It is easy to see that p iocos q. However, $f(p)$ $\mathsf{ioc\acute{o}s}$ $f(q)$. 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 $\mathsf{ioc\acute{o}s}$ $f(q)$ and 0 $\mathsf{ioc\acute{o}s}$ q (because $a? \notin \mathsf{ins}(0)$), this implies that $f(p)$ $\mathsf{ioc\acute{o}s}$ $f(q)$.

Example 14. *This example indicates that the use of output actions in negative premises of output-emitting rules would invalidate Theorem 21. Let f be defined by the following rules:*

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

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

$$b! \circlearrowright p \qquad\qquad b! \circlearrowright q \circlearrowright a!$$

where $b! \in O$. Again, p iocos q but $f(p)$ $\mathsf{ioc\acute{o}s}$ $f(q)$, because $f(p) \xrightarrow{a!} f(p)$ but $f(q) \xrightarrow{a!}\!\!\!\!\!\!/\ \ $.

Example 15. *This example indicates that the use of input actions in positive premises of output-emitting rules would invalidate Theorem 21. Let f be defined by the following rules:*

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

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

$$\delta! \circlearrowright p \circlearrowright b? \qquad\qquad q \circlearrowright \delta!$$

Again, p iocos q but $f(p)$ $\mathsf{ioc\acute{o}s}$ $f(q)$, because $f(p) \xrightarrow{a!} f(p)$ but $f(q) \xrightarrow{a!}\!\!\!\!\!\!/\ \ $.

7.2. Applying modal decomposition

Assume that we want to know whether $f(p_1, \ldots, p_n) \models \varphi$, with f specified by rules in iocoŝ-format and $\varphi \in \mathcal{L}_{\text{iocoŝ}}$. One could construct the LTS for $f(p_1, \ldots, p_n)$ from those for $p_1, \ldots, p_n$ and then use the rules defining the satisfaction relation $\models$ to check whether $f(p_1, \ldots, p_n)$ satisfies φ. However, as is well known, this approach suffers from the so-called state-explosion problem. Alternatively, one can apply a compositional approach: to check whether $f(p_1, \ldots, p_n) \models \varphi$, one first constructs, from φ and the rules defining the operation f, a collection of properties $\varphi_1, \ldots, \varphi_n$ such that $f(p_1, \ldots, p_n) \models \varphi$ if, and only if, $p_i \models \varphi_i$ for $i \in \{1, \ldots, n\}$; after that, one checks whether each statement $p_i \models \varphi_i$ holds. (Even though compositional model checking suffers from the 'formula-explosion problem' in the worst case and is therefore no panacea in general, variations on that approach have been applied successfully in the literature—see, for instance, [5, 29].)

Here we follow this compositional approach by applying the so-called modal decomposition method of [10, 17]. In those papers, ruloids play an important role. For a GSOS languages, a ruloid

$$\frac{\{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\} \cup \{x_i \xrightarrow{b_{ik}} \not\rightarrow \mid 1 \leq i \leq n, 1 \leq k \leq \ell_i\}}{D[\vec{x}] \xrightarrow{a} C[\vec{x}, \vec{y}]} \quad (3)$$

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, $D[\vec{x}]$ is a term with variables including at most the x_i's, $C[\vec{x}, \vec{y}]$ is a term 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.

In what follows, we will limit ourselves to considering only operations in iocoŝ-format specified by rules whose target is either a variable or a term of the form $g(z_1, \ldots, z_n)$, and the metavariables t and u will range over terms of those forms. The extension of our results to arbitrary rules in iocoŝ-format can be carried out along the lines in [10, 17].

In the light of our simplified setting, iocoŝ rules, i.e., GSOS rules in iocoŝ-format, play the role of ruloids in [10, 17]; the only other ruloids we need are those for variables, viz.

$$\frac{x \xrightarrow{a} x'}{x \xrightarrow{a} x'} \; a \in L.$$

In what follows, we often refer to ruloids as rules.

A crucial result from [9, 11] is that a transition $\sigma(t) \xrightarrow{a} p'$ is provable in a GSOS language if, and only if, there exist a substitution σ' and a GSOS ruloid $H/t \xrightarrow{a} u$ such that σ' satisfies all premises in H (denoted by $\sigma' \models H$), $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p'$.

Before defining the modal decomposition for $\mathcal{L}_{\text{iocoŝ}}$ in general, let us describe how to define it for the modal operator $\langle\!\langle a? \rangle\!\rangle$, with $a? \in I$. First, recall that $p \models \langle\!\langle a? \rangle\!\rangle \varphi$ iff either $p \xrightarrow{a?} \not\rightarrow$ or there exists some p' such that $p \xrightarrow{a?} p'$ with

$p' \models \varphi$. Hence, the modal decomposition must consider both cases: when is possible to apply a rule that gives rise to an $a?$-transition, and when it is not. This first case is dealt with following the definition of the decomposition of formulae in HML given in [17]. For the second case, we first define the following auxiliary notations and concepts.

Definition 17. *Let P be a GSOS language, t a term, and $a? \in I$ an input action. We write $R(t, a?)$ for the set of rules for t that emit $a?$. That is,*

$$R(t, a?) = \{r \in P \mid \exists H, u.\ r = H/t \xrightarrow{a?} u\}.$$

We write H_r for the set of premises of rule $r \in R(t, a?)$. Given a premise $\gamma \in H_r$ and a variable x, we define the formula $\mathsf{neg}(\gamma, x) \in \mathcal{L}_{\mathsf{iocos}}$ in the following way:

- $\mathsf{neg}(x \xrightarrow{b!}\!\!\!\!\!/\ , x) = \langle b! \rangle \mathbf{tt}$, *with* $b! \in O$.
- $\mathsf{neg}(x \xrightarrow{b?} x', x) = \langle\!\langle b? \rangle\!\rangle \mathbf{ff}$, *with* $b? \in I$.
- $\mathsf{neg}(y \xrightarrow{b!}\!\!\!\!\!/\ , x) = \mathsf{neg}(y \xrightarrow{b?} y', x) = \mathbf{tt}$, *with* $y \neq x$.

Finally, we write $\chi(t, a?)$ for the set of all the functions that pick a premise in H_r for each $r \in R(t, a?)$—that is,

$$\chi(t, a?) = \{\eta \mid \eta : R(t, a?) \to \bigcup_{r \in R(t, a?)} H_r, \text{ such that } \eta(r) \in H_r, \forall r \in R(t, a?)\}.$$

Remark 10. *Intuitively, $\mathsf{neg}(\gamma, x)$ gives us a iocos formula that captures that the premise γ is not satisfied. For example, if $\gamma = x \xrightarrow{b!}\!\!\!\!\!/\ $, then $\mathsf{neg}(x \xrightarrow{b!}\!\!\!\!\!/\ , x) = \langle b! \rangle \mathbf{tt}$ and given a closed substitution σ, if $\sigma(x) \models \langle b! \rangle \mathbf{tt}$, then $\sigma(x)$ does not satisfy the premise γ.*

The following example illustrates the discussion in Remark 10.

Example 16. *Let us consider $t = f(x, y)$, with rules*

$$r_1 \ \frac{x \xrightarrow{a!}\!\!\!\!\!/\ }{f(x,y) \xrightarrow{a?} x} \qquad\qquad r_2 \ \frac{x \xrightarrow{a?} x' \quad y \xrightarrow{b?} y'}{f(x,y) \xrightarrow{a?} x'}\ .$$

We want to characterise when $\sigma(f(x, y)) \xrightarrow{a?}\!\!\!\!\!/\ $, for each closed substitution σ. In order to do so, first let us define $\chi(f(x, y), a?)$. Since there are two premises in H_{r_2}, $\chi(f(x, y), a?) = \{\eta_1, \eta_2\}$, where:

$$\begin{aligned} \eta_1 &= \{r_1 \mapsto x \xrightarrow{a!}\!\!\!\!\!/\ , r_2 \mapsto x \xrightarrow{a?} x'\}, \\ \eta_2 &= \{r_1 \mapsto x \xrightarrow{a!}\!\!\!\!\!/\ , r_2 \mapsto y \xrightarrow{b?} y'\}. \end{aligned}$$

Now, let us define $\psi_{\eta_i}(z) = \bigwedge_{r \in R(t,a?)} \mathsf{neg}(\eta_i(r), z)$*, for* $z \in \{x, y\}$*. That is, working up to logical equivalence,*

$$\begin{array}{rclcrcl} \psi_{\eta_1}(x) & = & \langle a! \rangle \mathbf{tt} \wedge \langle\!\langle a? \rangle\!\rangle \mathbf{ff}\,, & & \psi_{\eta_1}(y) & = & \mathbf{tt}\,, \\ \psi_{\eta_2}(x) & = & \langle a! \rangle \mathbf{tt}\,, & & \psi_{\eta_2}(y) & = & \langle\!\langle b? \rangle\!\rangle \mathbf{ff}\,. \end{array}$$

We observe that if $\sigma(x) \models \psi_{\eta_i}(x)$ *and* $\sigma(y) \models \psi_{\eta_i}(y)$ *for some* $i \in \{1, 2\}$*, then* $\sigma(f(x, y)) \xrightarrow{a?}\!\!\!\!\!\!\diagup$*, and that the converse implication also holds. (See Proposition 26 in Appendix B for the formal result.)*

We are now ready to define modal decomposition à la Fokkink and van Glabbeek for the logic $\mathcal{L}_{\mathsf{iocos}}$. We will give the definition for an alternative syntax of the logic $\mathcal{L}_{\mathsf{iocos}}$ where, as noted in Definition 3, we use finitary conjunctions and disjunctions, and follow the convention that an empty conjunction stands for **tt** and an empty disjunction stands for **ff**.

Definition 18 (Modal decomposition)**.** *The decomposition function*

$$\cdot^{-1} : \mathbb{T}(\Sigma) \to (\mathcal{L}_{\mathsf{iocos}} \to \mathcal{P}(\mathsf{Var} \to \mathcal{L}_{\mathsf{iocos}}))$$

is defined in the following way:

- $\psi \in t^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *iff* $\psi \in t_\chi^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi) \cup t_R^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *where* $t_\chi^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *and* $t_R^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *are defined as follows:*
  - $\psi \in t_\chi^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *iff exists a function* $\eta \in \chi(t, a?)$ *such that* $\psi = \psi_\eta$*, where for each* $x \in \mathsf{Var}$*,*

$$\psi_\eta(x) = \bigwedge_{r \in R(t,a?)} \mathsf{neg}(\eta(r), x).$$

  - $\psi \in t_R^{-1}(\langle\!\langle a? \rangle\!\rangle \varphi)$ *iff there are a rule* $r = H/t \xrightarrow{a?} t' \in R(t, a?)$ *and a decomposition mapping* $\psi' \in u^{-1}(\varphi)$ *such that, for each* $x \in \mathsf{Var}$*,*

$$\psi(x) = \begin{cases} \bigwedge_{\substack{x \xrightarrow{b?} y \in H \\ y \in var(u)}} \langle\!\langle b? \rangle\!\rangle \psi'(y) \wedge \psi'(x) & \text{if } x \in var(u) \\ \bigwedge_{\substack{x \xrightarrow{b?} y \in H \\ y \in var(u)}} \langle\!\langle b? \rangle\!\rangle \psi'(y) & \text{if } x \notin var(u) \end{cases}$$

- $\psi \in t^{-1}(\langle a! \rangle \varphi)$ *iff there exist some rule* $H/t \xrightarrow{a!} u$ *and some decomposition mapping* $\psi' \in u^{-1}(\varphi)$ *such that, for each* $x \in \mathsf{Var}$*,*

$$\psi(x) = \begin{cases} \bigwedge_{\substack{x \xrightarrow{b!} y \in H \\ y \in var(u)}} \langle b! \rangle \psi'(y) \wedge \bigwedge_{x \xrightarrow{c?}\!\!\!\!\!\!\diagup \in H} \langle\!\langle c? \rangle\!\rangle \mathbf{ff} \wedge \psi'(x) & \text{if } x \in var(u) \\ \bigwedge_{\substack{x \xrightarrow{b!} y \in H \\ y \in var(u)}} \langle b! \rangle \psi'(y) \wedge \bigwedge_{x \xrightarrow{c?}\!\!\!\!\!\!\diagup \in H} \langle\!\langle c? \rangle\!\rangle \mathbf{ff} & \text{if } x \notin var(u) \end{cases}$$

- $\psi \in t^{-1}(\bigvee_{i \in I} \varphi_i)$ *iff* $\psi \in \bigcup_{i \in I} t^{-1}(\varphi_i)$ *(where* I *is a finite index set).*

- $\psi \in t^{-1}(\bigwedge_{i\in I}\varphi_i)$ *iff there are* $\psi_i \in t^{-1}(\varphi_i)$ *for* $i \in I$*, such that for all* $x \in \mathsf{Var}$*,* $\psi(x) = \bigwedge_{i\in I}\psi_i(x)$*, where* I *is finite.*

Remark 11. *The only positive premises that are considered when defining* $\psi(x)$ *are those for which* $x \xrightarrow{b?} y$ *with* $y \in var(u)$*. Indeed, by the definition of GSOS rules, since all the variables are distinct, if* $x \xrightarrow{b?} y \in H$*,* $\psi(y) = \mathbf{tt}$ *if* $y \notin var(u)$*. Also, since* $\langle\!\langle a?\rangle\!\rangle \mathbf{tt} \equiv \mathbf{tt}$*, the last sentence means that the only positive premises* $x \xrightarrow{b?} y$ *that give a non-trivial* $\psi(y)$ *are those such that* $y \in var(u)$*.*

Remark 12. *Notice that* $t^{-1}(\mathbf{ff}) = \emptyset$ *(case* $\bigvee_{i\in I}\varphi_i$ *for* $I = \emptyset$*) and* $t^{-1}(\mathbf{tt}) = \{\psi\}$*, where* $\psi(x) = \mathbf{tt}$ *for all* x *(case* $\bigwedge_{i\in I}\varphi_i$ *for* $I = \emptyset$*).*

Our reader may wonder why the definition of $\psi \in t_R^{-1}(\langle\!\langle a?\rangle\!\rangle\varphi)$ is sufficient to ensure that $\sigma(t) \models \langle\!\langle a?\rangle\!\rangle\varphi$ whenever $\sigma(x) \models \psi(x)$ for all $x \in var(t)$. Indeed, the definition in question does not explicitly require that $\sigma(x)$ satisfy the negative premises in H_r. (Observe that this cannot be done because there is no formula in $\mathcal{L}_{\mathsf{iocos}}$ that is satisfied by all the processes that cannot perform a $b!$-transition, for some output action $b!$.) However, as the proof of Theorem 22 will make clear (see Appendix B), since the rules are in iocos format, it isn't necessary to express the negative premises in the definition of $\psi \in t_R^{-1}(\langle\!\langle a?\rangle\!\rangle\varphi)$. In fact, if σ does not satisfy any of the mappings $\psi' \in t_\chi^{-1}(\langle\!\langle a?\rangle\!\rangle\varphi)$, then σ satisfies all the premises of at least one rule r in $R(t, a?)$ and some mapping ψ associated to that rule contains all the information that is needed to determine whether $\sigma(t) \models \langle\!\langle a?\rangle\!\rangle\varphi$.

All this is formalised in Theorem 22, but first we will show some examples of the use of the modal decomposition.

Example 17. *Let us consider the following rules in the* iocos*-format (omitting the rules for the quiescence action):*

$$r_1 \ \frac{}{g(y) \xrightarrow{b!} g(y)} \qquad\qquad r_2 \ \frac{x \xrightarrow{a?} y \quad x \xrightarrow{b!}\!\!\!\!\!\!\rightarrow}{f(x) \xrightarrow{a?} g(y)} \ ,$$

where $a? \in I$ *and* $b! \in O$*. Let us calculate* $f(x)^{-1}(\langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt})$*. First, we have that* $R(f(x), a?) = \{r_2\}$ *and, since* H_{r_2} *has two premises, there are two functions* $\chi(t, a?)$*, namely* $\eta_1(r_2) = x \xrightarrow{a?} y$ *and* $\eta_2(r_2) = x \xrightarrow{b!}\!\!\!\!\!\!\rightarrow$*. Second, trivially,* $g(y)^{-1}(\langle b!\rangle\mathbf{tt}) = \{\phi\}$*, where* $\phi(x) = \mathbf{tt}$ *for all* x*.*

Now, let us write both $f(x)_\chi^{-1}(\langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt})$ *and* $f(x)_R^{-1}(\langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt})$*.*

- $\psi_{\eta_1}, \psi_{\eta_2} \in f(x)_\chi^{-1}(\langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt})$*, where* $\psi_{\eta_1}(x) = \langle\!\langle a?\rangle\!\rangle\mathbf{ff}$*,* $\psi_{\eta_2}(x) = \langle b!\rangle\mathbf{tt}$ *and* $\psi_{\eta_1}(z) = \psi_{\eta_2}(z) = \mathbf{tt}$ *for all* $z \neq x$*.*
- $\psi \in f(x)_R^{-1}(\langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt})$ *iff* $\psi(x) = \langle\!\langle a?\rangle\!\rangle\mathbf{tt} \equiv \mathbf{tt}$ *and* $\psi(z) = \mathbf{tt}$ *for all* $z \neq x$*.*

This means that any process p *is such that* $f(p) \models \langle\!\langle a?\rangle\!\rangle\langle b!\rangle\mathbf{tt}$*. Indeed, let us assume that* $p \xrightarrow{a?}$ *and* $p \xrightarrow{b!}\!\!\!\!\!\!\rightarrow$*, in this case* $f(p) \xrightarrow{a?} \sigma(g(y))$ *and we are done. On the other hand, if either* $p \xrightarrow{a?}\!\!\!\!\!\!\rightarrow$ *or* $p \xrightarrow{b!}$*, then* $f(p) \xrightarrow{a?}\!\!\!\!\!\!\rightarrow$ *and therefore* $f(p)$ *trivially satisfies the formula.*

Theorem 22. *Let P be a GSOS language in* iocos-*format. For each term t, formula φ and closed substitution σ, we have $\sigma(t) \models \varphi$ iff there exists $\psi \in t^{-1}(\varphi)$ such that $\sigma(x) \models \psi(x)$, for all $x \in var(t)$.*

Proof. The proof of this result can be found in Appendix B. □

The next counter-example illustrates why the iocos-format is needed in Theorem 22.

Example 18. *Let f be defined by the following rules, which are not in* iocos-*format and are taken from Example 11 on page 27:*

$$r_1 \ \frac{x \xrightarrow{a?} y}{f(x) \xrightarrow{a?} 0} \qquad\qquad r_2 \ \frac{}{f(x) \xrightarrow{a?} f(x)} \ .$$

Applying the modal decomposition method to obtain $f(x)^{-1}(\langle\!\langle a?\rangle\!\rangle\langle\!\langle a?\rangle\!\rangle\mathbf{ff})$, it is not hard to see that $\psi_{r_1} \in f(x)^{-1}_R(\langle\!\langle a?\rangle\!\rangle\langle\!\langle a?\rangle\!\rangle\mathbf{ff})$, where $\psi_{r_1}(x) = \mathbf{tt}$. The process s, with $s \xrightarrow{\delta!} s$, satisfies $\mathbf{tt} = \psi_{r_1}(x)$. On the other hand, $f(s) \not\models \langle\!\langle a?\rangle\!\rangle\langle\!\langle a?\rangle\!\rangle\mathbf{ff}$, which means that the decomposition method is incorrect for this operation.

The rules for f do not meet clause 2a in Definition 16 because $a?$ belongs to the positive trigger for x in r_1 but not to that for x in r_2. This clause is essential in the proof of the "if implication" in Theorem 22.

7.3. 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 $\vec{p}$ in an LTS, should satisfy the following property:

$$f(\vec{p}) \xrightarrow{\delta!} p' \text{ iff } p' = f(\vec{p}) \text{ and, for each } a! \in O \setminus \{\delta!\},\ f(\vec{p}) \not\xrightarrow{a!} . \qquad (4)$$

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

Definition 19. *We say that the following sets of formulae contradict each other:*

- $\{x \xrightarrow{a} y\}$ *and* $\{x \not\xrightarrow{a}\}$ *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* $H \cup H' = \{x \not\xrightarrow{b!} \mid b! \in O\}$.

Formulae $x \xrightarrow{a} y$ and $x \not\xrightarrow{a}$ 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 $p \xrightarrow{b!}$ for each $b! \in O$. This observation motivates the third requirement in Definition 19.

Definition 20. *We say that an operation f is* quiescent consistent *if the set of rules for f satisfies the following two constraints:*

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

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

$$\frac{\{l_1, \ldots, l_n\}}{f(\vec{x}) \xrightarrow{\delta!} f(\vec{x})},$$

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 23. *If f is quiescent consistent then Property (4) holds for f.*

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

We end the section showing a negative result for the (binary) merge operator of Example 4. In that example, we mentioned that, as noted in [6], the merge of two systems need not satisfy Property (4). Here we prove that there is no extension of the $\delta!$-emitting rules for the merge operator with rules in iocos-format that leads to an operation affording Property (4). The interested reader can find examples of operators that are both quiescent and iocos conforming in the paper [23].

Proposition 24. *There is no extension of the rules for the (binary) merge operator with a collection of $\delta!$-emitting rules in* iocos*-format that guarantees consistent quiescent behaviour.*

Proof. Our goal is to add $\delta!$-emitting rules in iocos-format to those for the merge operator so that, for all p, q

$$p \wedge q \xrightarrow{\delta!} r \Leftrightarrow r = p \wedge q \text{ and } p \wedge q \xrightarrow{a!} \forall a! \in O \setminus \{\delta!\} \,. \tag{5}$$

We will show that this is impossible. Let us recall the output emitting rules for the binary merge operator:

$$r_{a!}\ \frac{x \xrightarrow{a!} x' \quad y \xrightarrow{a!} y'}{x \wedge y \xrightarrow{a!} x' \wedge y'}\ a! \in O.$$

In particular, r_δ ensures that $p \wedge q \xrightarrow{\delta!} p \wedge q$ for all quiescent p and q.

Now, assume that there is a set of $\delta!$-emitting rules R in iocos-format such that $R \cup \{r_{a!} \mid a! \in O\}$ ensures Property (5). Let us consider $a! \xrightarrow{a!} 0$ and $b! \xrightarrow{b!} 0$, with $a \neq b$. Since $a! \wedge b! \not\xrightarrow{c!}$ for each $c! \in O \setminus \{\delta!\}$ there are some rule $r = H/x \wedge y \xrightarrow{\delta!} t$ and closed substitution σ such that: (i) $\sigma(x) = a!$, $\sigma(y) = b!$; (ii) $\sigma \models H$; and (iii) $\sigma(t) = a! \wedge b!$.

Since r is in iocos-format and $\sigma \models H$, its positive premises can only have the form $x \xrightarrow{a!} x'$ for some x' and $y \xrightarrow{b!} y'$ for some y'. The negative premises of H, if any, have the form $x \not\xrightarrow{c?}$ or $y \not\xrightarrow{c?}$ for some $c? \in I$. Moreover, as $a! \xrightarrow{a!} 0$, $b! \xrightarrow{b!} 0$ and $\sigma(t) = a! \wedge b!$, none of the variables x' and y' can occur in t. So t can only have the form $x \wedge y$, $a! \wedge y$, $x \wedge b!$ or $a! \wedge b!$. We claim that it must be $x \wedge y$. Indeed, assume, by way of example, that $t = a! \wedge y$. Consider now the substitution $\sigma'(x) = a! + b!$ and $\sigma'(y) = b!$, where $a! + b! \xrightarrow{a!} 0$ and $a! + b! \xrightarrow{b!} 0$. It is straightforward to check that $\sigma' \models H$. Hence, r yields $(a! + b!) \wedge b! \xrightarrow{\delta!} a! \wedge b!$, which contradicts Property (5). Similar examples can be constructed if $t = x \wedge b!$ and $t = a! \wedge b!$.

Thus, we have $r = H/x \wedge y \xrightarrow{\delta!} x \wedge y$. Consider now the substitution ρ such that $\rho(x) = \rho(y) = a! + b!$ and $\rho(z) = 0$ for $z \notin \{x, y\}$. This substitution satisfies all the positive premises in H and, since all the negative premises in H are of the form $y \not\xrightarrow{c?}$ for some $c? \in I$, ρ also satisfies those. Hence $\rho \models H$, and ρ together with r proves $(a! + b!) \wedge (a! + b!) \xrightarrow{\delta!} (a! + b!) \wedge (a! + b!)$.

On the other hand, using r_a, we also have that $(a! + b!) \wedge (a! + b!) \xrightarrow{a!} 0 \wedge 0$, which contradicts Property (5).

It follows that no extension of the rules for the (binary) merge operator with $\delta!$-emitting rules in iocos-format satisfies Property (5). □

8. Conclusion

In this paper, we have further developed the theory of iocos [20, 21, 22] by studying logical characterisations of this relation and its compositionality. We have also compared the proposed logical characterisation of iocos with an existing logical characterisation for ioco proposed by Beohar and Mousavi. The article also offers a precongruence rule format for iocos 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.

We have provided a connection between the precongruence rule format for iocos and logical characterisations for that relation by establishing a modal

decomposition result, which yields a compositional model-checking procedure for the studied logic with respect to systems described using operations in iocos-format. We have also studied fixed-point extensions of the logics introduced in this paper and offered a characteristic-formula construction for processes modulo iocos.

In future research, it would be interesting to study whether the precongruence rule format we provide in this paper can be made more general and whether it can be derived from a modal decomposition result in the style of Fokkink and van Glabbeek. Our modal decomposition result relies heavily on properties of the iocos format; we believe therefore that generalizing the precongruence rule format might be difficult.

In this paper, we have considered the theory of iocos as presented in [20, 21, 22, 23]. That theory does not consider the internal action τ, which is the source of many of the complications in the classic ioco testing theory. Extending the theory of iocos to a setting with the internal action τ is an interesting avenue for future research. As a first step, one could develop a version of iocos with the action $\tau \notin L$, but without abstracting from internal steps in system executions. This can be done by requiring that condition 3 in the definition of an iocos-relation (Definition 2) hold also with respect to τ-transitions—that is, that internal transitions performed by implementations should be matched by specifications as in the classic simulation preorder [34]. In this framework, we could replay the theory developed in this paper. By way of example, the rule format given in Definition 16 can be extended to the resulting version of iocos by requiring that τ-emitting rules have only input actions as labels of negative premises and output actions as labels of positive premises. The resulting rule format would be powerful enough to express the hiding operation considered in [43]. Of course, the ultimate goal of extending iocos with internal actions would be to develop a theory that abstracts from τ like ioco and that has some of the pleasing properties of that classic notion of conformance, including a test generation procedure. We consider the study of such a theory and of its potential applications a worthy research goal for the future.

[1] Fides Aarts and Frits W. Vaandrager. Learning I/O automata. In Paul Gastin and François Laroussinie, editors, *21st International Conference on Concurrency Theory, CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 2010.

[2] Luca Aceto, Ignacio Fábregas, Carlos Gregorio-Rodríguez, and Anna Ingólfsdóttir. Logical characterisations and compositionality of input-output conformance simulation. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2017.

[3] Luca Aceto, Wan Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, chapter 3, pages 197–292. Elsevier, 2001.

[4] Luca Aceto, Anna Ingólfsdóttir, Paul Blain Levy, and Joshua Sack. Characteristic formulae for fixed-point semantics: a general framework. *Mathematical Structures in Computer Science*, 22(2):125–173, 2012.

[5] Henrik Reif Andersen. Partial model checking (extended abstract). In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science*, pages 398–407. IEEE Computer Society, 1995.

[6] 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.

[7] Harsh Beohar and Mohammad Reza Mousavi. Two logical characterizations for input-output conformance. In Preproceedings of EXPRESS/SOS'14 (Short paper), July 2014.

[8] 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.

[9] Bard Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages*. PhD thesis, Massachusetts Institute of Technolog, 1993.

[10] Bard Bloom, Wan Fokkink, and Rob J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Compututational Logic*, 5(1):26–78, 2004.

[11] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.

[12] Bard Bloom and Albert R. Meyer. Experimenting with process equivalence. *Theoretical Computer Science*, 101(2):223–237, 1992.

[13] Rance Cleaveland and Matthew Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5(1):1–20, 1993.

[14] Sjoerd Cranen, Jan Friso Groote, Jeroen J.A. Keiren, Frank P.M. Stappers, Erik P. de Vink, Wieger Wesselink, and Tim A.C. Willemse. An overview of the mCRL2 toolset and its recent advances. In Nir Piterman and Scott A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2013.

[15] 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.

[16] Ignacio Fábregas, David de Frutos-Escrig, and Miguel Palomino. Logics for contravariant simulations. In John Hatcliff and Elena Zucca, editors, *Formal Techniques for Distributed Systems, FMOODS 2010 and FORTE 2010. Proceedings*, volume 6117 of *Lecture Notes in Computer Science*, pages 224–231. Springer, 2010.

[17] Wan Fokkink, Rob J. van Glabbeek, and Paulien de Wind. Compositionality of Hennessy-Milner logic by structural operational semantics. *Theoretical Computer Science*, 354(3):421–440, 2006.

[18] Rob J. van Glabbeek. The linear time – branching time spectrum I: The semantics of concrete, sequential processes. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.

[19] Susanne Graf and Joseph Sifakis. A modal characterization of observational congruence on finite terms of CCS. In *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, volume 172 of *Lecture Notes in Computer Science*, pages 222–234. Springer, 1984.

[20] 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.

[21] 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.

[22] 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.

[23] Carlos Gregorio-Rodríguez, Luis Llana, and Rafael Martínez. An axiomatic semantics for iocos conformance relation. *Journal of Logical and Algebraic Methods in Programming*, 100:152–184, 2018.

[24] Jan Friso Groote and Mohammad Reza Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press, 2014.

[25] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[26] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[27] Jeroen J.A. Keiren. The modal μ-calculus (version 1.1). *Technical report*, pages 1–19, 2013.

[28] Bartek Klin and Beata Nachyla. Some undecidable properties of SOS specifications. *Journal of Logical and Algebraic Methods in Programming*, 87:94–109, 2017.

[29] François Laroussinie and Kim Guldstrand Larsen. CMC: A tool for compositional model-checking of real-time systems. In Stanislaw Budkowski, Ana R. Cavalli, and Elie Najm, editors, *Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE XI / PSTV XVIII'98, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII)*, volume 135 of *IFIP Conference Proceedings*, pages 439–456. Kluwer, 1998.

[30] Kim Guldstrand Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72(2&3):265–288, 1990.

[31] Kim Guldstrand Larsen and Xinxin Liu. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, 1991.

[32] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines—a survey. *Proceedings of the IEEE*, 84(8):1090–1123, August 1996.

[33] 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.

[34] Robin Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489. William Kaufmann, 1971.

[35] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[36] Neda Noroozi, Mohammad Reza Mousavi, and Tim A. C. Willemse. On the complexity of input output conformance testing. In José Luiz Fiadeiro, Zhiming Liu, and Jinyun Xue, editors, *Formal Aspects of Component Software - 10th International Symposium, FACS 2013*, volume 8348 of *Lecture Notes in Computer Science*, pages 291–309. Springer, 2013.

[37] Bernhard Steffen and Anna Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Compututation*, 110(1):149–163, 1994.

[38] Colin Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.

[39] 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, *Proceedings 7th Workshop on Model-Based Testing*, volume 80 of *EPTCS*, pages 73–87, 2012.

[40] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[41] Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.

[42] 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.

[43] 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.

Appendix A. Proof of Theorem 21

Remark 13. *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 iocos and such that: if $p_i \,\mathcal{S}\, q_i$, for each $1 \leq i \leq n$, and f is an n-ary operation in iocos-format 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 25. *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 $\sigma(t) \,\mathcal{S}\, \rho(t)$.*

Proof of Theorem 21. We show that $\mathcal{S}$ is an iocos simulation by induction on the definition of $\mathcal{S}$. This is clear if $p \,\mathcal{S}\, q$ because $p \,\mathsf{iocos}\, q$. 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 relations 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 (2) on page 23 and 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
  - $q_i \xrightarrow{b_{ik}} \!\!\!\!\!/\;\;$ 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 16, 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
  - $p_i \xrightarrow{b_{ik}} \!\!\!\!\!/\;\;$, 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 $r = \dfrac{H}{f(x_1, \ldots, x_n) \xrightarrow{a?} t}$ of the form (2) on page 23 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 (2) on page 23 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 16 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 25. 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 16), 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 (2) on page 23 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 $1 \leq j \leq m_i$,
  - $p_i \xrightarrow{b_{ik}}\!\!\!\!\!\!/\;\;$ for each $1 \leq i \leq n$ and $1 \leq k \leq \ell_i$, and
  - $\sigma(C[\vec{x}, \vec{y}]) = p'$.

  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 16 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, $p_i \xrightarrow{b_{ik}}\!\!\!\!\!\!/\;\;$ $(1 \leq i \leq n$ 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

$$\begin{array}{rcll} \rho(x_i) & = & q_i & (1 \leq i \leq n), \text{ and} \\ \rho(y_{ij}) & = & q_{ij} & (1 \leq i \leq n, 1 \leq j \leq m_i) \end{array}$$

yields the transition

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

By construction, $\sigma(z)\mathcal{S}\rho(z)$ for each variable z occurring in $C[\vec{x}, \vec{y}]$. Therefore Lemma 25 now yields that

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

and we are done. □

Appendix B. Proof of Theorem 22

Before proving Theorem 22, we will show two auxiliary lemmas that will be use in its proof.

Lemma 26. *Let P be a GSOS language in* iocos*-format. For every term t, input $a? \in I$ and closed substitution σ, we have that $\sigma(t) \xrightarrow{a?}\!\!\!\!\!\!\not\;\;$ if, and only if, there exists some $\eta \in \chi(t, a?)$ such that $\sigma(x) \models \psi_\eta(x)$ for each $x \in var(t)$.*

Proof. The result follows straightforwardly. First, recall that, by definition, $\psi_\eta(x) = \bigwedge_{r \in R(t,a?)} \mathsf{neg}(\eta(r), x)$. Second, note that given a premise $\eta(r) \in H_r$, the formula $\mathsf{neg}(\eta(r), x)$ is such that $\sigma(x) \models \mathsf{neg}(\eta(r), x)$ iff $\sigma(x)$ does not satisfy premise $\eta(r)$. So, if there is some $\eta \in \chi(t, a?)$ such that $\sigma(x) \models \psi_\eta(x)$, for each $x \in var(t)$, then no rule in $R(t, a?)$ can be used to derive a transition from $\sigma(t)$. Conversely, if $\sigma(t) \xrightarrow{a?}\!\!\!\!\!\!\not\;\;$, then, for each $r \in R(t, a?)$, there is some premise in H_r that is not satisfied by $\sigma(t)$. Let η be the choice function that picks those premises. Then, by the discussion above, $\sigma(x) \models \psi_\eta(x)$ for each $x \in var(t)$. □

Lemma 27. *Let P be a GSOS language in* iocos*-format. Let $r = H/t \xrightarrow{a?} u$ and $r' = H'/t \xrightarrow{a?} u$ be such that if $x \xrightarrow{b?} y \in H'$ with $y \in var(u)$, then $x \xrightarrow{b?} y \in H$. Consider $\psi, \psi' \in t_R^{-1}(\langle\!|a?|\!\rangle\varphi)$ and $\phi \in u^{-1}(\varphi)$ such that ψ is obtained using ruloid r and ϕ, and ψ' is obtained using ruloid r' and ϕ. Then $\psi(x)$ implies $\psi'(x)$ for all $x \in var(t)$.*

Proof. We show that each conjunct of $\psi'(x)$ is implied by $\psi(x)$. Without loss of generality, let us assume $x \in var(u)$. By construction,

- $\psi(x) = \bigwedge_{\substack{x \xrightarrow{b?} y \in H \\ y \in var(u)}} \langle\!|b?|\!\rangle \phi(y) \wedge \phi(x)$.
- $\psi'(x) = \bigwedge_{\substack{x \xrightarrow{c?} z \in H' \\ z \in var(u)}} \langle\!|c?|\!\rangle \phi(z) \wedge \phi(x)$.

By hypothesis, if $x \xrightarrow{c?} z \in H'$ with $z \in var(u)$, then $x \xrightarrow{c?} z \in H$. This implies that each conjunct $\langle\!|c?|\!\rangle\phi(z)$ of $\psi'(x)$ is also a conjunct of $\psi(x)$, and the claim follows. □

Proof of Theorem 22. The proof is by structural induction on the formula ϕ. We only show the case of $\phi = \langle\!|a?|\!\rangle\varphi$ since the others are the same as in [17].

Only if implication: Let us suppose that $\sigma(t) \models \langle\!|a?|\!\rangle\varphi$. Then, by definition of the satisfaction relation $\models$, either $\sigma(t) \xrightarrow{a?}\!\!\!\!\!\!/\;\;$ or there exists some p such that $\sigma(t) \xrightarrow{a?} p$ and $p \models \varphi$.

Let us first consider the case when $\sigma(t) \xrightarrow{a?}\!\!\!\!\!\!/\;\;$. This means that it is not possible to apply any of the rules with conclusion $t \xrightarrow{a?} u$ for some u. By Lemma 26, there exist $\eta \in \chi(t, a?)$ and ψ_η such that $\sigma(x) \models \psi_\eta(x)$ for each $x \in var(t)$. That is, there exists some $\psi_\eta \in t^{-1}_\chi(\langle\!|a?|\!\rangle\varphi)$ such that $\sigma(x) \models \psi_\eta(x)$, for all $x \in var(t)$, and we are done.

On the other hand, if there exists some p such that $\sigma(t) \xrightarrow{a?} p$ and $p \models \varphi$, there are a rule $r = H/t \xrightarrow{a?} u$ and a closed substitution σ' such that $\sigma' \models H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$. Since $\sigma'(u) \models \varphi$ by the induction hypothesis there exists some $\phi \in u^{-1}(\varphi)$ such that $\sigma'(z) \models \phi(z)$ for all $z \in var(u)$. Now, let us consider $\psi \in t^{-1}_R(\langle\!|a?|\!\rangle\varphi)$ constructed as in Definition 18 from the rule $r = H/t \xrightarrow{a?} u$ and $\phi \in u^{-1}(\varphi)$.

In this case we have that $\sigma(x) = \sigma'(x) \xrightarrow{b?} \sigma'(y)$ and $\sigma(x) = \sigma'(x) \xrightarrow{c!}\!\!\!\!\!\!/\;\;$, for all $x \xrightarrow{b?} y \in H$ and $x \xrightarrow{c!}\!\!\!\!\!\!/\;\; \in H$. We show that $\sigma(x) = \sigma'(x) \models \psi(x)$ for all $x \in var(t)$. We distinguish two cases depending on whether $x \in var(u)$ or not.

- If $x \in var(u)$, by definition $\psi(x) = \bigwedge_{x \xrightarrow{b?} y \in H, y \in var(u)} \langle\!|b?|\!\rangle\phi(y) \wedge \phi(x)$, and by the inductive hypothesis $\sigma'(x) \models \phi(x)$ and $\sigma'(y) \models \phi(y)$, for each $y \in var(u)$. Since $\sigma'(x) \xrightarrow{b?} \sigma'(y)$, because σ' satisfies H, we obtain the following: $\sigma'(x) \models \bigwedge_{x \xrightarrow{b?} y \in H, y \in var(u)} \langle\!|b?|\!\rangle\phi(y)$. Hence, $\sigma(x) = \sigma'(x) \models \psi(x)$.
- If $x \notin var(u)$, by definition that $\psi(x) = \bigwedge_{x \xrightarrow{b?} y \in H, y \in var(u)} \langle\!|b|\!\rangle\phi(y)$, and using the same reasoning as before, we conclude $\sigma(x) = \sigma'(x) \models \psi(x)$.

Finally, for all $x \in var(t)$, $\sigma(x) = \sigma'(x) \models \psi(x)$.

If implication: Let us suppose that $t = f(x_1, \ldots, x_n)$ and that there is some $\psi \in t^{-1}(\langle\!|a?|\!\rangle\varphi)$ such that $\sigma(x) \models \psi(x)$ for all $x \in var(t)$. We have to show that $\sigma(t) \models \langle\!|a?|\!\rangle\varphi$.

Since the claim holds if $\sigma(t) \xrightarrow{a?}\!\!\!\!\!\!/\;\;$, we need only consider the case that $\sigma(t) \xrightarrow{a?}$. This means that there are a rule $r = H/t \xrightarrow{a?} u$ and a substitution σ_r such that (a) $\sigma_r \models H$, and (b) $\sigma(t) = \sigma_r(t)$, that is, $\sigma(x) = \sigma_r(x)$ for all $x \in var(t)$. Since $\sigma(x) \models \psi(x)$ for all $x \in var(t)$, by Lemma 26, item (a) above yields that $\psi \in t^{-1}_R(\langle\!|a?|\!\rangle\varphi)$.

Suppose that ψ is constructed using a rule $r' = H'/t \xrightarrow{a?} u'$ and $\phi \in u'(\varphi)$. Since we assume that the GSOS language is in iocos-format, condition (2) in Definition 16, tells us that there is a rule $r'' = H''/t \xrightarrow{a?} u$ such that:

(*i*) for all $i \in \{1, \ldots, n\}$, the positive trigger for variable x_i in r'' is included in the positive trigger for variable x_i in r;

(*ii*) for all $i \in \{1, \ldots, n\}$, the negative trigger for variable x_i in r'' is included in the negative trigger for variable x_i in r; and

(*iii*) if $x_i \xrightarrow{b?} z \in H''$ and $z \in var(u)$, then $x_i \xrightarrow{b?} z$ is also contained in H'.

Let $\psi' \in t_R^{-1}(\langle\!|a?|\!\rangle\varphi)$ be constructed from r'' and $\phi \in u'^{-1}(\varphi)$. By Lemma 27, $\psi(x) \Rightarrow \psi'(x)$ for each $x \in var(t)$. Therefore, $\sigma(x) = \sigma_r(x) \models \psi'(x)$, for all $x \in var(t)$.

Our goal is to use rule r'' to show that $\sigma(t) \xrightarrow{a?} p$ for some p such that $p \models \varphi$. To this end, we will construct a substitution σ'' with the following properties: (1) $\sigma''(t) = \sigma(t)$, (2) $\sigma''(x) \models H''$, (3) $\sigma''(x) \models \phi(x)$ for $x \in var(u')$. We note that using σ'' and rule r'', we then have, by (1) and (2), that $\sigma(t) \xrightarrow{a?} \sigma''(u')$ and $\sigma''(u') \models \varphi$ by (3) and the inductive hypothesis. So to complete the proof we are left to construct σ''.

We start by setting $\sigma''(x) = \sigma(x)$ for each $x \in var(t)$. This ensures (1). Moreover, since $\sigma_r(x) = \sigma(x) = \sigma''(x)$ for each $x \in var(t)$ and σ_r satisfies all the negative premises in H by item (a), by condition *ii* σ'' satisfies all the negative premises in H''. We now show how to extend the definition of σ'' in order to satisfy also the positive premises in H''.

- If $x_i \xrightarrow{b?} y \in H''$ and $y \notin var(u')$, then by condition (i) above there is a positive premise $x_i \xrightarrow{b?} y' \in H$. Since $\sigma_r(x_i) = \sigma(x_i) = \sigma''(x_i)$ and $\sigma_r \models H$, there is some p' such that $\sigma_r(x_i) \xrightarrow{b?} p'$. We set $\sigma''(y) = p'$.

- If $x_i \xrightarrow{b?} y \in H''$ and $y \in var(u')$, then $\sigma''(x_i) = \sigma_r(x_i) = \sigma(x_i) \models \psi'(x)$. Now, by construction, $\psi'(x)$ has a conjunct $\langle\!|b?|\!\rangle\phi(y)$. Again, by condition (i, there is some p' such that $\sigma_r(x_i) \xrightarrow{b?} p' \models \phi(y)$. We can therefore set $\sigma''(y) = p'$ to guarantee item (3) above for y.

Finally, note that if $x \in var(t) \cap var(u')$, $\psi'(x)$ has $\phi(x)$ as conjunct and therefore setting $\sigma''(x) = \sigma(x)$ guarantees (3) for those variables.

This completes the definition of σ'' having properties (1)–(3), and we are done. □

Appendix C. Proof of Theorem 23

We start by establishing the following lemma, which states the correctness of the definition of contradictory sets of formulae in Definition 19.

Lemma 28. *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 19.

- Assume that $H_1' = \{x \xrightarrow{a} y\}$ and $H_2' = \{x \xrightarrow{a}\!\!\!\!/\ \}$ for some $a \in L$. Since $\sigma(x) \xrightarrow{a} \sigma(y)$, there is no substitution ρ such that $\rho(x) = \sigma(x) \xrightarrow{a}\!\!\!\!/$.
- 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 $H_1' \cup H_2' = \{x \xrightarrow{b!}\!\!\!\!/\ \mid b! \in O\}$ 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 23. We will show the two implications separately.

Proof of Theorem 23. We assume that f is quiescent consistent and prove Property (4):

$$f(\vec{p}) \xrightarrow{\delta!} p' \text{ iff } p' = f(\vec{p}) \text{ and, for each } a! \in O \setminus \{\delta!\},\ f(\vec{p}) \xrightarrow{a!}\!\!\!\!/\ .$$

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

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

and a substitution σ such that $\sigma(f(\vec{x})) = f(\vec{p})$, $\sigma(t) = p'$ and σ satisfies the premises in H. By condition $[\delta_1]$ in Definition 20, we have that $t = f(\vec{y})$ for some vector of variables $\vec{y}$ 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 $p' = \sigma(t) = \sigma(f(\vec{y})) = f(\vec{p})$. We are therefore left to show that $f(\vec{p}) \xrightarrow{b!}\!\!\!\!/$ for each $b! \in O \setminus \{\delta!\}$. To this end, it is enough to prove that if

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

is a rule for f, then no substitution ρ that agrees with σ over $\vec{x}$ can satisfy H'. In order to see this, observe that, by condition $[\delta_1]$ in Definition 20, the sets H and H' are contradictory. Since σ satisfies H, by Lemma 28, no substitution ρ

that agrees with σ over $\vec{x}$ can satisfy H'.

If implication: Assume now that $f(\vec{p}) \stackrel{b!}{\not\longrightarrow}$ for all $b! \in O \setminus \{\delta!\}$. We will argue that $f(\vec{p}) \stackrel{\delta!}{\longrightarrow} f(\vec{p})$. 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 $\vec{x}$ that p_j does not satisfy. For each $1 \leq i \leq n$, we define $\bar{l}_i$ as follows:

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

Here the variables y_i's are all different and distinct from the variables in $\vec{x}$. By condition $[\delta_2]$ in Definition 20, we have that the set of rules for f includes the rule

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

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 $\rho(f(\vec{x})) = f(\vec{p})$ and ρ satisfies $\{\bar{l}_1, \ldots \bar{l}_n\}$. Hence, we conclude that $f(\vec{p}) \stackrel{\delta!}{\longrightarrow} f(\vec{p})$, and we are done. □