Theoretical Computer Science 410 (2009) 4044–4066

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Reactive systems, (semi-)saturated semantics and coalgebras on presheaves

Filippo Bonchi [a,*], Ugo Montanari [b,1]

[a] CWI, Science Park 123, XG Amsterdam, Netherlands
[b] Dipartimento di Informatica, Universita' di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy

ARTICLE INFO

Keywords:
Behavioral equivalences
Reactive systems
Logic programming
Petri nets

ABSTRACT

The semantics of process calculi has traditionally been specified by labelled transition systems (LTSs), but, with the development of name calculi, it turned out that reaction rules (i.e., unlabelled transition rules) are often more natural. This leads to the question of how behavioral equivalences (bisimilarity, trace equivalence, etc.) defined for LTS can be transferred to unlabelled transition systems. Recently, in order to answer this question, several proposals have been made with the aim of automatically deriving an LTS from reaction rules in such a way that the resulting equivalences are congruences. Furthermore, these equivalences should agree with the standard semantics, whenever one exists.

In this paper, we propose saturated semantics, based on a weaker notion of observation and orthogonal to all the previous proposals, and we demonstrate the appropriateness of our semantics by means of two examples: logic programming and open Petri nets. We also show that saturated semantics can be efficiently characterized through the so called semi-saturated games. Finally, we provide coalgebraic models relying on presheaves.

1. Introduction

The operational semantics of process calculi is usually given in terms of transition systems labelled with actions, which, when visible, represent both observations and interactions with the external world. The abstract semantics is given in terms of behavioral equivalences, which depend on the action labels and on the amount of branching structure considered. Behavioral equivalences are often congruences with respect to the operations of the language, and this property expresses the compositionality of the abstract semantics.

A simpler approach, inspired by classical formalisms like λ-calculus, Petri nets, term and graph rewriting, and pioneered by the Chemical Abstract Machine [5], defines operational semantics by means of *structural axioms* and *reaction rules*. Process calculi representing complex systems – in particular those able to generate and communicate names – are often defined in this way, since structural axioms give a clear idea of the intended structure of the states, while reaction rules, which are often non-conditional, give a direct account of the possible steps. Transitions caused by reaction rules, however, are not labelled, since they represent evolutions of the system without interactions with the external world. Thus reduction semantics in itself is neither abstract nor compositional. To enhance the expressiveness of reduction semantics, Leifer and Milner proposed, in [33], a systematic method for deriving bisimulation congruences from reduction rules. The main idea is the following: a process p can do a move with label $c[-]$ and become p' iff $c[p] \rightsquigarrow p'$. This definition was inspired by the work of Sewell

* Corresponding author. Tel.: +31 20 592 9333; fax: +31 20 592 4199.
E-mail addresses: fibonchi@di.unipi.it (F. Bonchi), ugo@di.unipi.it (U. Montanari).
1 Tel.: +39 050 2212721; fax: +39 050 2212726.

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.06.010

[51]. Also, the approach of observing contexts imposed on agents at each step was introduced in [43], yielding the notion of *dynamic bisimilarity*.

Leifer and Milner also introduced the categorical notions of relative pushout (RPO) and idem relative pushout (IPO) in order to specify a/the minimal context that allows the state to react with a given rule. This construction leads to labelled transition systems (LTS) that use only contexts generated by IPOs, and not all contexts, as labels, and thus are smaller than in the latter case. Bisimilarity, trace equivalence and failure equivalence on this LTS (called IPO semantics) are congruences under rather restrictive conditions.

In this paper our aim is, as in the ordinary case, to derive a bisimilarity congruence from given reduction rules. However, we introduce in the transition system *all* context-labelled transitions which make a state and a rule match. We call the resulting equivalences *saturated*. Saturated equivalences are coarser than IPO ones and have nice properties, e.g., they are always congruences, but the LTS is infinite-branching in more cases. Here we develop a *semi-saturated* technique that allows one to compute saturated equivalences without considering all matching contexts. In fact, if we call Alice the player choosing the move and Bob the player choosing a matching reply, we prove that if Alice chooses an IPO move and Bob replies with any matching move, the resulting equivalence is the saturated one, even if the moves to be considered are usually much less.

Moreover, we show that in some relevant cases saturated equivalences are exactly what we want, while IPO equivalences are too fine. In the paper we discuss two important cases: *logic programming* and *open Petri nets*.

We model logic programming in a way similar to [15]. It turns out that saturated trace congruence coincides with the ordinary logic semantics of logic programming, while the IPO trace congruence yields a finer semantics, know in the logic programming community as *S*-semantics [22]. Interestingly enough, a goal (i.e., a conjunction of atomic goals) and the head of a clause must adapt in two different ways: both must be instantiated, but in addition the head must be ($\wedge$-)composed with other formulas which stay idle in the reduction. We are able to obtain both adaptations at the same time within our approach, without resorting to an infinite number of rules, as it is usually the case for the ordinary construction, since agents are normally forced to be closed.

Open Petri nets [30,4] are an interactive extension of P/T nets. They can interact with each other by exchanging tokens through *open places*, i.e., those places that are visible from the environment. We model open Petri nets as a special kind of multiset rewriting systems, where multisets of tokens can be added only into open places. Again, while IPO bisimilarity yields a finer semantics, the saturated bisimilarity coincides with the semantics previously proposed in [4].

In the last part of the paper, we introduce coalgebraic models for IPO and saturated bisimilarity. Universal Coalgebra [47] provides a categorical framework where abstract semantics of interactive systems are described as morphisms to their minimal representatives. More precisely, given an endofunctor $\mathbf{B}$ on a category $\mathbf{C}$, a coalgebra is an arrow $\alpha\colon X \to \mathbf{B}(X)$ of $\mathbf{C}$ and a coalgebra morphism from α to β is an arrow $h\colon X \to X'$ of $\mathbf{C}$ with $h\,;\beta = \alpha\,;\mathbf{B}(h)$. Under certain conditions on $\mathbf{C}$ and $\mathbf{B}$, a category of coalgebras admits a final object. Ordinary labeled transition systems (with finite or countable branching) can be represented as coalgebras with final object for a suitable functor on **Set**. Then, in order to prove that two states are equivalent, we have to check if they are identified by the final morphism, and the image of the given coalgebra through the latter is the minimal representative.

However, this representation of interactive systems forgets about the algebraic structure, which is usually very relevant in practical cases, since compositionality is the key to master complexity. In particular, the property that bisimilarity respects the operations (i.e., that it is a congruence, which is essential for making abstract semantics compositional) is not reflected in the structure of the model.

In [54], *bialgebras* are introduced as a model with both algebraic and coalgebraic structure, while a related approach based on *structured coalgebras* is presented in [19]. In this paper we will use coalgebras on a category of *presheaves* $\mathbf{Set}^{\mathbf{C}}$, where objects of $\mathbf{C}$ represent *interfaces* of systems and arrows represent *environment*, i.e., (unary) contexts in which systems can be embedded. Coalgebras on presheaves have been widely used in order to model nominal process calculi [53,24,23] but with a slightly different flavor. In these works, the index category only models the names of systems: usually, objects are sets of names and arrows are (injective) names substitutions. Here instead, the index category $\mathbf{C}$ is a Lawvere-like category where arrows are algebraic contexts. By using such kind of categories, standard results of the theory of coalgebras guarantee the compositionality of bisimilarity.

This paper is an extended version of [11,12]. In the former, we have introduced saturated semantics and the semi-saturated game by using, as examples, logic programming and a fragment of open π-calculus [48] that is not presented here, since it could be more conveniently modeled via an extended framework that we have introduced in [13]. In [12], we have introduced coalgebraic models for IPO and saturated semantics by means of structured coalgebras. The coalgebraic models based on presheaves that we will show in this paper are essentially the same as the structured coalgebraic ones of [12], but they are presented in a new perspective that highlights the links between coalgebras on presheaves and reactive systems. Most of the examples come from the Ph.D. thesis of the first author [6] and they have never been published.

2. The theory of reactive systems

In this section, we summarize the theory of reactive systems proposed in [33] to derive labelled transition systems (LTSs) and bisimulation congruences from a given reaction semantics. The theory is centered on the concepts of *term*, *context* and *reaction rules*: contexts are arrows of a category, terms are arrows having as source 0 (a special object that denotes "no

holes"), and reaction rules are pairs of terms. Hereafter, given a category $\mathbf{C}$, $|\mathbf{C}|$ denotes the class of its objects, $||\mathbf{C}||$ the class of its arrows and $\mathbf{C}[i, j]$ (with $i, j \in |\mathbf{C}|$) the class of arrows having source i and target j.

Definition 1 (*Reactive System*). A *reactive system* $\mathcal{R}$ consists of:

1. a category $\mathbf{C}$,
2. a distinguished object $0 \in |\mathbf{C}|$,
3. a composition-reflecting subcategory $\mathbf{D}$ of *reactive contexts*,
4. a set of pairs $\mathfrak{R} \subseteq \bigcup_{i \in |\mathbf{C}|} \mathbf{C}[0, i] \times \mathbf{C}[0, i]$ of *reaction rules*.

The reactive contexts are those in which a reaction can occur. By composition-reflecting we mean that $d; d' \in \mathbf{D}$ implies $d, d' \in \mathbf{D}$. Note that the rules have to be ground, i.e., left-hand and right-hand sides have to be terms without holes and, moreover, with the same codomain.

From reaction rules, one generates the reaction relation by closing them under all reactive contexts. Formally the *reaction relation* is defined by taking $p \rightsquigarrow q$ if there is $\langle l, r\rangle \in \mathcal{R}$ and $d \in \mathbf{D}$ such that $p = l; d$ and $q = r; d$.

Definition 2 (*Free Lawvere Theory and Free Term Category [32]*). Let Σ be a signature. The *free Lawvere theory* for Σ, denoted as $\mathbf{Th}[\Sigma]$, is a category with object natural numbers and morphisms $c : m \to n$ being n-tuples of m-holed terms. Composition is substitution of terms, and identities $id_n : n \to n$ are $\langle -_1, -_2, \ldots, -_n\rangle$. An arrow $t : m \to n$ of $\mathbf{Th}[\Sigma]$ is *linear* if each of the m holes appears exactly once in t. The *free term category* of Σ, denoted as $\mathbf{C}_\Sigma$, is the subcategory of $\mathbf{Th}[\Sigma]$ having the same objects, but only linear arrows.

During the whole paper, we will think of the base category $\mathbf{C}$ as a Lawvere-like category (i.e., a category where arrows represent terms and contexts of a signature). Thus we will often refer to an arrow $c \in ||\mathbf{C}||$ as to a unary context $c[-]$, and to the composition of arrows $c; d$ as to $d[c[-]]$.

Example 1 (*Term Rewriting*). A term rewriting system consists of a signature Σ and a set of rules $\mathfrak{R}$ of the form $l \to r$, where l and r are terms of Σ. The operational semantics is simply obtained by contextualizing and instantiating l and r. In other words, $p \rightsquigarrow q$ if and only if there exist $l \to r \in \mathfrak{R}$ such that there exist a context $c[-]$ and an instantiation i, such that $p = c[l[i]]$ and $q = c[r[i]]$.

In order to see a term rewriting system as a reactive system, we have to restrict our attention to *ground term rewriting systems*, i.e., those where the rules are *ground* (namely, these cannot be further instantiated). Every ground term rewriting system defines a reactive system where $\mathbf{C}_\Sigma$ is the base category, the distinguished object is the natural number 0, all the contexts are reactive (i.e., $\mathbf{D} = \mathbf{C}_\Sigma$) and the set of rules is $\mathfrak{R}$.

Definition 3 (*Quotiented Lawvere Theory and Quotiented Term Category*). Let Σ be a signature and E be a congruence relation over the arrows of $\mathbf{Th}[\Sigma]$. The Lawvere theory quotiented by E is the category $\mathbf{Th}[\Sigma/\mathbf{E}]$ having the same objects as $\mathbf{Th}[\Sigma]$, but arrows $[p] : m \to n$ are equivalence classes of arrows $p : m \to n$ of $\mathbf{Th}[\Sigma]$. It is easy to verify that the composition operator is well defined and $\mathbf{Th}[\Sigma/\mathbf{E}]$ is still a category. In the same way, we can define the term category quotiented by E, denoted as $\mathbf{C}^{\mathbf{E}}_\Sigma$.

Example 2 (*Simple Process Calculus (SPC)* [51,52]). Consider the following fragment of CCS for a set of channels name $\mathcal{N}$.

$$p, q = \mathbf{0} \;\mid\; a \;\mid\; \bar{a} \;\mid\; p \mid q \quad a \in \mathcal{N}.$$

The signature Σ consists of a set of input and output constants parametrized over $\mathcal{N}$, the null process $\mathbf{0}$, and the binary operator of parallel composition $|$. Processes are considered up to *structural congruence* $\equiv$ which is the smallest congruence that ensures associativity, commutativity and identity (w.r.t. $\mathbf{0}$) of the parallel operator. The intuitive operational semantics is that a process sending on a channel named $a \in \mathcal{N}$ and a process receiving on the same channel, can react and disappear. In symbols $\bar{a} \mid a \rightsquigarrow \mathbf{0}$.

The reactive system of simple process calculus is $\mathcal{SPC} = \langle \mathbf{C}^{\equiv}_\Sigma, 0, \mathbf{C}^{\equiv}_\Sigma, \mathfrak{R}\rangle$ where $\mathbf{C}^{\equiv}_\Sigma$ is the term category over Σ quotiented by $\equiv$, the object 0 is the natural number, all contexts are reactive and the set of rules $\mathfrak{R}$ is $\{\langle \bar{a} \mid a, \mathbf{0}\rangle$ s.t. $a \in \mathcal{N}\}$.

The operational behavior of a reactive system is expressed as an unlabelled transition system. On the other hand, many useful behavioral equivalences are only defined for LTSs. In order to obtain an LTS, we can plug a term p into some context $c[-]$ and observe if a reaction occurs. In this case, we have that $p \xrightarrow{c}$. Categorically speaking, this means that $p; c$ matches $l; d$ for some rule $\langle l, r\rangle \in \mathcal{R}$ and some reactive context d. This situation is formally depicted by diagram (i) in Fig. 1: a commuting diagram like this is called a *redex square*. Formally, a redex square is four arrows p, c, l, d such that $p; c = l; d, d \in \mathbf{D}$ and, for some r, $\langle l, r\rangle \in \mathfrak{R}$.

Definition 4 (*Saturated Transition System*). The *saturated transition system* (SATTS for short) is the LTS having as states arrows with source 0 and transitions are defined as $p \xrightarrow{c}_{SAT} q$ iff $c[p] \rightsquigarrow q$.

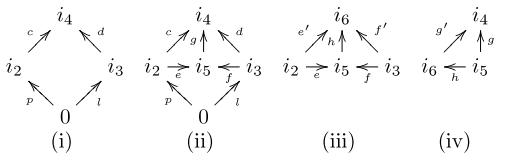


Fig. 1. Redex Square and RPO.

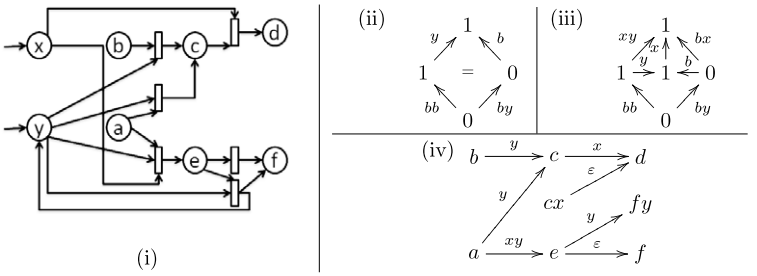


Fig. 2. (i) An open input Petri net; (ii) $bb \xrightarrow{y}_{SAT} bc$; (iii) $bb \xrightarrow{xy}_{SAT} bcx$; (iv) The IPOTS of a, b and cx.

Note that SATTS is often infinite-branching since all contexts that allow reactions may occur as labels. Another problem of SATTS is that it has redundant transitions. For example, consider the term a of SPC. The observer can put this term into the context $\overline{a} \mid -$ and observe a reaction. This corresponds to the transition $a \xrightarrow{\overline{a}|-}_{SAT} 0|0$. However we also have $a \xrightarrow{p|\overline{a}|-}_{SAT} p \mid 0 \mid 0$ as a transition, yet p does not contribute to the reaction. Hence we need a notion of "minimal context that allows a reaction". Leifer and Milner define idem pushouts (IPOs) to capture this notion.

Definition 5 (*RPO*). Let the diagrams in Fig. 1 be in some category **C**. Let (i) be a commuting diagram. Any tuple $\langle i_5, e, f, g\rangle$ which makes (ii) commute is called a *candidate* for (i). A *relative pushout (RPO)* is the smallest such candidate. More formally, it satisfies the universal property that, given any other candidate $\langle i_6, e', f', g'\rangle$, there exists a unique mediating morphism $h : i_5 \to i_6$ such that (iii) and (iv) commute.

Definition 6 (*IPO*). A commuting square like diagram (i) of Fig. 1 is called *idem pushout (IPO)* if $\langle i_4, c, d, id_{i_4}\rangle$ is its RPO.

Definition 7 (*Redex RPOs*). A reactive system *has redex RPOs* if every redex square has an RPO.

Definition 8 (*IPO Transition System*). The *IPO transition system* (IPOTS for short) is the LTS having as states arrows with source 0 and transitions are defined as $p \xrightarrow{c}_I r$; d iff $d \in \mathbf{D}$, $\langle l, r\rangle \in \mathcal{R}$ and diagram (i) in Fig. 1 is an IPO.

In other words, if inserting p into the context $c[-]$ matches l; d, and $c[-]$ is the "smallest" such context (according to the IPO condition), then p transforms to r; d with label $c[-]$, where r is the reduct of l.

Bisimilarity on IPOTS is referred to as *IPO bisimilarity* (denoted by $\sim^I$), and Leifer and Milner have shown that if the reactive system has redex RPOs, then it is a congruence (i.e., it is preserved under all contexts).

Theorem 1 (*From [33]*). *If $\mathcal{R}$ has redex-RPOs, then $\sim^I$ is a congruence.*

It can be easily shown that bisimilarity over SATTS is always a congruence, whether or not the underlying category has redex RPOs. In this paper, we will focus on this bisimilarity, which will be called *saturated bisimilarity* (denoted by $\sim^S$).

Proposition 1. *In all reactive systems, $\sim^S$ is a congruence.*

Example 3 (*Running Example: Open Input Petri Net*). Given a set X, we write $X^{\oplus}$ for the free commutative monoid over X. A multiset $m \in X^{\oplus}$ is a function from X to ω (the set of natural numbers) that associates a multiplicity to every element of X. Given two multisets m_1 and m_2, $m_1 \oplus m_2$ is defined as $\forall x \in X$, $m_1 \oplus m_2(x) = m_1(x) + m_2(x)$. We write $\emptyset$ and ε to denote, respectively, the empty set and the empty multiset. In order to make the notation lighter, we will use aab to denote the multiset $\{a, a, b\}$. Sometimes we will use $a^n b^m$ to denote the multisets containing n copies of a and m copies of b.

An *open input Petri net* [4] (open net, for short) is $N = \langle S, T, pre, post, IP\rangle$ where S is the set of places, $IP \subseteq S$ is the set of input places, T is the set of transitions (with $S \cap T = \emptyset$), $pre, post : T \to S^{\oplus}$ are functions mapping each transition to its pre- and post-set[2]. A *marking* m over a net N is a multiset of tokens over the places of N, i.e., $m \in S^{\oplus}$. A *marked net* is an open net N together with a marking m.

Fig. 2(i) shows an open net where, as usual, circles represents places and rectangles transitions. Arrows from places to transitions represent *pre*, while arrows from transitions to places represent *post*. Ingoing arrows without source denote input places. Thus in Fig. 2(i) the input places are x and y.

[2] Open input Petri nets are a subclass of *Open nets* [4]. Besides input places, open nets have also output places. We consider only input places in order to make the example simpler.

The operational semantics of marked nets is expressed by the following rules, where we use ${}^\bullet t$ and $t^\bullet$ to denote, respectively, $pre(t)$ and $post(t)$.

$$\frac{t \in T}{N, {}^\bullet t \oplus c \xrightarrow{\tau} N, t^\bullet \oplus c} \qquad \frac{i \in IP^\oplus}{N, m \xrightarrow{+i} N, m \oplus i}$$

The transitions generated by the leftmost rules are the standard transitions of P/T nets, while the rightmost rule allows the environment to insert tokens into input places. The abstract semantics of open nets is defined in [4] as the standard bisimilarity over such LTS.

Given an open input Petri net $N = \langle S, T, pre, post, IP\rangle$, we can define the corresponding reactive system as $\mathcal{N} = \langle \mathbf{OPN}_N, 0, \mathbf{OPN}_N, \mathfrak{T}\rangle$, where $\mathbf{OPN}_N$ is defined as follows:

- 0 and 1 are the only objects,
- arrows $0 \to 0$ and $0 \to 1$ are multisets on S, while arrows $1 \to 1$ are multisets on IP,
- identities are the empty multisets and composition is the union of multiset.

The set of rules $\mathfrak{T}$ is $\{\langle {}^\bullet t, t^\bullet\rangle : 0 \to 0 \mid t \in T\}$. Intuitively, each transition of the net defines a reaction rule as a pair of arrows with source and target 0. Instead, arrows from 0 to 1 model the states of the net (i.e., multisets over all the places), while arrows from 1 to 1 model contexts (i.e., multisets over input places).

It is easy to see that (when restricting to $\mathbf{OPN}_N[0, 1]$) the reaction relation ($\rightsquigarrow$) for $\mathcal{N}$ coincides with the τ-transitions defined by the ordinary operational semantics. For saturated transition instead, look at Fig. 2(ii). It describes the transition $bb \xrightarrow{y}_{SAT} bc$. The rule is the pair of arrows $\langle by, c\rangle : 0 \to 0$ (corresponding to the left-topmost transition of Fig. 2(i)). This rule is contextualized with $b : 0 \to 1$ (that is a token in a closed place), while the state $bb : 0 \to 1$ can be contextualized only with tokens in input places. Note that the state bb can use only the rule $\langle by, c\rangle$. Indeed all the other rules contain a, c, e in the left hand side and tokens in those places cannot be added to $bb : 0 \to 1$.

However, the saturated transition system for bb is infinite branching, since $bb \xrightarrow{x^i y^{j+1}}_{SAT} bcx^i y^j$ for all $i, j \in \omega$. As an example, look at the outer square of Fig. 2(iii). For this reason, instead of considering all possible contexts, we want to consider only the minimal ones. By applying the definition of IPO to the category $\mathbf{OPN}_N$, we get that the IPOTS (restricted to $\mathbf{OPN}_N[0, 1]$) coincides with the one generated by the following rule.

$$\frac{t \in T \quad m = (m \cap^\bullet t) \oplus c \quad i \in IP^\oplus \qquad {}^\bullet t = (m \cap^\bullet t) \oplus i}{N, m \xrightarrow{i}_I N, t^\bullet \oplus c}$$

Fig. 2(iv) shows the IPOTS of multisets a, b and cx. This labeled transition system is finite because it takes into account only the minimal contexts. Now, reconsider the multiset bb. We have that $bb \xrightarrow{y}_I bc$, but $bb \not\xrightarrow{xy}_I bcx$. Indeed, in Fig. 2, diagram (i) is an IPO, while diagram (ii) is not, since the multiset x is not really needed to perform the transition.

It is worth noting that $\mathbf{OPN}_N$ has RPOs, while it does not have pushouts. For example there is no pushout for the arrows $a : 0 \to 1$ and $by : 0 \to 0$.

3. Suitability of IPO and saturated semantics

After their introduction, several attempts have been made to encode several specification formalisms (Petri nets [38, 34,49], Mobile Ambients [9,10], CCS [39,8], λ-calculus [40,20], asynchronous π-calculus [29], fusion calculus [27], etc.) as reactive systems, either hoping to recover the standard observational equivalences, whenever such a behavioral semantics exists, or trying to distill a meaningful new semantics. Unfortunately, IPO semantics is often too fine-grained. As shown in [10], IPO bisimilarity for Mobile Ambients is strictly contained in the one proposed in [35]. In [11], we showed that, for the open π-calculus, IPO bisimilarity is strictly included into open bisimilarity. For the case of CCS, IPO bisimilarity is strictly included in the ordinary bisimilarity for a bigraphical encoding [39], while the two coincide when encoding CCS into ordinary graphs [8]. In [29], it is shown that IPO bisimilarity coincides with the ordinary semantics of a summation-free fragment of the asynchronous π-calculus. However, we conjecture that extending the encoding to processes with summation, the two semantics do not coincide anymore. Indeed, in our opinion, the axiom $a?x.(a!x \mid p) + \tau.p \sim \tau.p$ is hard to capture with IPO bisimilarity.

Also, for open nets, IPO bisimilarity is too strict.

Example 4 ($\sim^I$ *is Too Strict in Open Input Petri Nets*). Consider the IPOTS of the multisets e and cx of the net N shown in Fig. 2(iv). The former can interact both with the rule $\langle e, f\rangle$ generating the transition $e \xrightarrow{\varepsilon}_I f$ and with the rule $\langle ey, fy\rangle$ generating the transition $e \xrightarrow{y}_I fy$. The latter can interact only with the rule $\langle cx, d\rangle$ generating the transition $cx \xrightarrow{\varepsilon}_I d$. Thus $e \not\sim^I cx$, but they are equivalent according to the standard abstract semantics (Example 3). Intuitively, e and cx cannot be distinguished by an external observer that can insert tokens into input places and observe if some reaction occurs. Moreover, $e \sim^S cx$. Indeed, when e proposes the SATTS move $e \xrightarrow{y}_{SAT} fy$, cx can answer with $cx \xrightarrow{y}_{SAT} dy$ and $fy \sim^S dy$ since both cannot move.

Now consider the multiset a and b. We have that $a \nsim_{IPO} b$ but they are bisimilar according to the ordinary semantics, and moreover $a \sim^S b$. Indeed when a proposes $a \xrightarrow{xy}_{SAT} e$, b can answer with $b \xrightarrow{xy}_{SAT} cx$ and, as shown above, $e \sim^S cx$.

The above examples show that $\sim^I$ is strictly finer than the standard semantics for nets. It is easy to prove that the latter coincides with $\sim^S$ (restricted to the homset $\mathbf{OPN}_N[0, 1]$).

In the above example, saturated semantics coincides with ordinary semantics. However, when considering full process calculi equipped with recursion, saturated semantics are often too coarse. For example, the CCS processes $\omega = \tau.\Omega$ and $\Theta = \tau.\Omega + a.\Omega$ are saturated bisimilar [42], yet not strong bisimilar. This problem becomes potentially serious when considering *weak semantics*. Intuitively, two systems are saturated bisimilar if they cannot be distinguished by an external observer that, in any moment of their execution, can insert them into some context and observe a reduction. However, since in weak semantics, reductions cannot be observed, all systems are equivalent.

This argument suggests to us that, in general terms, saturated semantics should be equipped with some basic observations on the states, in the style of *barbs* [42]. In [10], the first author together with Gadducci and Monreale, has shown that for Mobile Ambients [16], IPO bisimilarity is still too strict, while saturated bisimilarity (properly extended with barbs) coincides with the bisimilarity defined in [45]. Moreover, a weak variant of saturated bisimilarity with barbs coincides with the one defined in [35].

In Section 5, we will show that, in Logic Programming, saturated trace equivalence (extended with an observation ϕ) coincides with the ordinary *logic equivalence* of Logic Programming.

In the next section, we will introduce semi-saturated semantics that allows us to "efficiently characterize" saturated semantics. The idea is presented for the case of bisimilarity (using, as a running example, open nets), but it is very general, and can be employed also for barbed bisimilarity (as shown in [16]).

4. Semi-saturated semantics

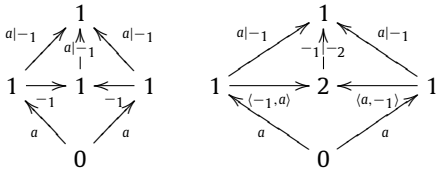
In this section, we introduce *semi-saturated game*: a general technique that allows one to *efficiently* characterize saturated semantics. By efficiently, we mean that we avoid considering the whole saturated transition system (SATTS) that is usually too big, since it is labeled with all possible contexts. Instead of the SATTS, we use the IPO transition system (IPOTS), whose labels are just the minimal contexts that allow some reaction. However, we do not consider the usual abstract semantics over the IPOTS but a slightly refined version of them: if we call Alice the player choosing the move and Bob the player choosing a matching reply, when Alice chooses an IPO move, Bob can reply with a move from SATTS.

We will prove that semi-saturated semantics coincide with saturated semantics whenever the reactive system has redex IPOs.

Definition 9 (*Redex IPOs*). A reactive system *has redex IPOs*, if every redex square has at least one IPO as candidate.

Clearly this constraint is weaker than having redex RPOs (Definition 7) that is required by Theorem 1. Having RPOs means to have a minimum candidate (i.e., a candidate smaller than all the others), while having IPOs allows one to have several minimal candidates (also not comparable among them). The following example shows the difference between redex IPOs and redex RPOs.

Example 5 (*IPOs in Simple Process Calculus*). Recall the category $\mathbf{C}_\Sigma^{\equiv}$ of Example 2. This is the term category of a signature having some constants and a binary operator that is associative, commutative and with identity. This category does not posses RPOs: consider the exterior squares in diagrams (i) and (ii) below (note that they are equal). This square has no RPOs since it has, as candidates, the arrows inside which are not comparable (in the sense that neither is smaller than the other). But note that both the candidates are IPOs, since they have, as candidates, only isomorphic diagrams.



In this section we will show semi-saturated games for both bisimilarity (Section 4.1) and a generalization of trace equivalence (Section 4.2).

4.1. Semi-saturated and symbolic bisimilarity

Here we introduce two alternative and, in some cases, finitary characterization of saturated bisimilarity and we will prove that they coincide with $\sim^S$.

Definition 10 (*Semi-saturated Bisimulation*). A symmetric relation R is a *semi-saturated bisimulation* if and only if whenever $p\,R\,q$,

- if $p \xrightarrow{c}_I p'$ then $q \xrightarrow{c}_{SAT} q'$ and $p' \, R \, q'$.

We call the union of all semi-saturated bisimulations *semi-saturated bisimilarity* (denoted by $\sim_{SS}$).

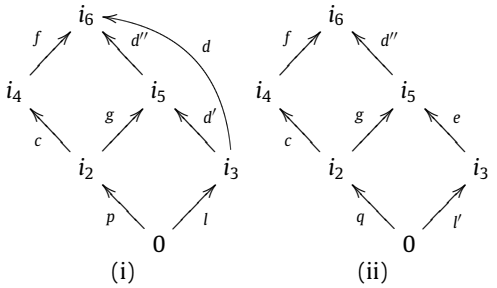
Theorem 2 states that, in the presence of redex IPOs, this kind of bisimilarity coincides with saturated bisimilarity (and thus it is a congruence). In this way, we can prove that two processes are saturated bisimilar just starting with IPO moves. Once an IPO move is chosen, the context $c[-]$ is fixed, and thus only the $\rightsquigarrow$ moves from $c[q]$ must be considered. Leifer and Milner have shown that $\sim^I$ is a congruence if the reactive system has redex RPOs, i.e., if for each redex-square there exists an RPO. For $\sim^S$ it is sufficient to require that the reactive system has redex IPOs.

Theorem 2. *If $\mathcal{R}$ has redex-IPOs, then semi-saturated bisimilarity coincides with saturated bisimilarity (i.e., $p \sim^{SS} q \iff p \sim^S q$).*

Proof. We prove that $\sim^{SS} \subseteq \sim^S$, showing that the contextual closure S of semi-saturated bisimilarity

$$S = \{\langle c[p], c[q]\rangle \mid p \sim^{SS} q, \ c \in \mathbf{C}\}$$

is a saturated bisimulation.



Suppose that $c[p] \xrightarrow{f}_{SAT} p'$. Then for some $\langle l, r\rangle \in \mathfrak{R}$ and $d \in \mathbf{D}$ we have that the exterior square of diagram (i) commutes and $p' = d[r]$. Since $\mathcal{R}$ has redex IPOs we are able to construct an IPO as the inner square of diagram (i) and then $p \xrightarrow{g}_I d'[r]$. Since $p \sim^{SS} q$ we have that $q \xrightarrow{g}_{SAT} e[r']$ for some $e \in \mathbf{D}$ and $\langle l', r'\rangle \in \mathfrak{R}$ with $d'[r] \sim^{SS} e[r']$. Now we can put the upper square of diagram (i) on the redex square generating this transition and we obtain diagram (ii) that trivially commutes. Hence $c[q] \xrightarrow{f}_{SAT} d''[e[r']]$, and $(p', d''[e[r']]) \in S$ because $p' = d[r] = d''[d'[r]]$ and $d'[r] \sim^{SS} e[r']$.

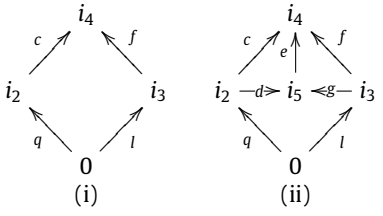
To prove that $\sim^S \subseteq \sim^{SS}$ it is sufficient to observe that if $p \xrightarrow{a}_I p'$ then $p \xrightarrow{a}_{SAT} p'$. □

The above theorem allows one to recover saturated bisimilarity without considering all the transitions of SATTS, i.e., all possible contexts that allow some reaction. The following definition offers an alternative characterization of semi-saturated bisimulations: when Alice propose an IPO move labeled with c, Bob can reply with another IPO move labeled with a contexts d smaller than c.

Definition 11 (*Symbolic Bisimulation*). A symmetric relation R is a *symbolic bisimulation* if and only if whenever $p \, R \, q$,

- if $p \xrightarrow{c}_I p'$ then $\exists d \in \mathbf{C}, e \in \mathbf{D}$ such that $d; e = c$, $q \xrightarrow{d}_I q'$ and $p' \, R \, q'; e$.

Theorem 3. *If $\mathcal{R}$ has redex-IPOs, then a symmetric relation R is a semi-saturated bisimulation if and only if it is a symbolic bisimulation.*



Proof. Suppose that R is a semi-saturated bisimulation. Let p, q be processes such that $p \, R \, q$. Then, $p \xrightarrow{c}_I p'$ implies that $q \xrightarrow{c}_{SAT} q'$and $p' \, R \, q'$. Then, by definition of $\rightarrow_{SAT}$, there exists a redex square like diagram (i) where $q' = r; f$. Since the reactive system has redex IPOs, then there exists an IPO candidate like that in (ii). Note that both g and e are reactive contexts, since f is reactive. Then $q \xrightarrow{d}_I r; g$. Now note that $p' \, R \, q' = (r; g); e$.

The inverse implication is immediate by Theorem 2. □

It is worth noting that the definition does not require that the arriving states p' and q' are bisimilar. Indeed, it requires that p' is bisimilar to $q'; e$, that is the process q' inserted into the context e that is missing to d to equate c. This reminds us of several abstract semantics of different process calculi. Among these, symbolic open bisimilarity [48], asynchronous bisimilarity [2], efficient bisimilarity for explicit fusion [55] and large bisimilarity [3]. The links between these abstract semantics and the above definition have been exploited in [13].

Example 6 ($\sim^{SS}$ *in Open Input Petri Nets*). Recall the input net in Fig. 2(i). In Example 4 we have informally shown that $a \sim^{S} b$ and $e \sim^{S} cx$. Here we formally prove it by showing that

$$R = \{(a, b),\ (b, a),\ (c, c),\ (e, cx),\ (cx, e),\ (d, f),\ (f, d),\ (dy, fy),\ (fy, dy)\}$$

is a symbolic bisimulation. Consider the IPOTS of marking a, b and cx in Fig. 2(iv).

We can prove that R is a symbolic bisimulation just using $\rightarrow_I$. As an example consider the pair (a, b). When $a \xrightarrow{y}_I c$ then $b \xrightarrow{y}_I c$ and $c\,R\,c$. In this case the arrow d of Theorem 3 is y and e is the identity. When $a \xrightarrow{xy}_I e$ then $b \xrightarrow{y}_I c$ and $e\,R\,cx$. In this case the arrow d of Theorem 3 is y and e is x. For all the other pairs, we can proceed analogously.

Alternatively, we can show that R is a semi-saturated bisimulation. Consider (a, b). When $a \xrightarrow{y}_I c$ then $b \xrightarrow{y}_{SAT} c$ and $c\,R\,c$. When $a \xrightarrow{xy}_I e$ then $b \xrightarrow{xy}_{SAT} cx$ and $e\,R\,cx$.

4.2. Semi-saturated trace equivalences

In this subsection we introduce ϕ-trace equivalence, an abstract semantics that is parametric w.r.t. a predicate ϕ and that generalizes canonical trace equivalence. In the theory of reactive system, this semantics is not be considered yet. We are introducing it, because it will be relevant in Section 5 as abstract semantics of logic program. All the proofs are in Appendix A.

As in the case of bisimilarity, we define saturated and IPO ϕ-trace equivalence. The former is always a congruence, while the latter only when there exists *redex and context RPOs*. Moreover, we will introduce a semi-saturated version of it and we prove that this coincides with the saturated one, whenever the system has *redex and context IPOs*.

Definition 12 (ϕ-*trace Equivalence*). Let X be a set of states, L a set of labels and $\rightarrow \subseteq X \times L \times X$ a transition relation. Let $-;- : L \times L \rightarrow L$ be an associative operator on labels and let ϕ be a property on X. We say that $p, q \in X$ are ϕ-trace equivalent ($p \simeq^{\phi} q$) if the following conditions hold:

- $\phi(p)$ if and only if $\phi(q)$,
- if $p \overset{l}{\twoheadrightarrow} p' \wedge \phi(p')$ then $q \overset{l}{\twoheadrightarrow} q' \wedge \phi(q')$,
- if $q \overset{l}{\twoheadrightarrow} q' \wedge \phi(q')$ then $p \overset{l}{\twoheadrightarrow} p' \wedge \phi(p')$,

where $p \overset{l}{\twoheadrightarrow} p'$ iff $p \xrightarrow{l_1} p_2 \ldots p_n \xrightarrow{l_n} p'$ and $l = l_1; l_2; \ldots; l_n$ with $n \geq 1$.

Note that the above definition generalizes the notion of trace equivalence: when ϕ holds in every state of X and ; is string concatenation, then we have the classical trace semantics for $\rightarrow$.

In the rest of this section we will study this equivalence in the setting of reactive systems, and we will fix the ; operator to be context composition. As we did for bisimilarity, we can define this equivalence on the IPOTS (*IPO ϕ-trace equivalence* denoted by $\simeq^{\phi}_{I}$) or on the SATTS (*saturated ϕ-trace equivalence* denoted by $\simeq^{\phi}_{SAT}$).

In order to obtain a congruence, we have to require the following conditions:

1. ϕ is defined on all arrows, and the arrows satisfying ϕ form a composition-reflecting subcategory;
2. all contexts are reactive.

The first requirement is not very strong, and we will show that, in our encoding of logic programming, it holds. The second constraint is rather restrictive, but there are many formalisms for which it holds, as, for example, term rewriting (Example 1), DPO graph rewriting [21], logic programming (Section 5) and open input Petri nets (Example 3).

Proposition 2. *In a reactive system where all contexts are reactive and ϕ defines a composition-reflecting subcategory, $\simeq^{\phi}_{SAT}$ is a congruence.*

IPO bisimilarity is a congruence under the constraint of having all redex RPOs, while here IPO ϕ-trace equivalence is a congruence under the assumption that RPOs exist not only for redex squares but also for squares where the four arrows are contexts. We say that a reactive system has *redex and context RPOs* if it satisfies this constraint. We have to require this condition since we are working with the transitive closure of $\rightarrow_I$. A similar condition is needed in [14] where the authors require one to have all RPOs, in order to show that weak bisimulation is a congruence.

Let us explain what we mean by *contexts*. When defining the encoding of a formalism into a reactive system, one is interested only in some arrows of the base category that represent the state of the formalism. For example, in the reactive system for open nets $\mathcal{N}$, only the arrows of $\mathbf{OPN}_N$ of type $0 \rightarrow 1$ represent the state of the open net. In the case of logic

programming (Section 5) only arrows of type $\mathfrak{p} \to \mathfrak{t}^n$ represent logic formulas. By *contexts* we mean only those arrows of the base category that can be be post-composed with such arrows. In the case of open nets, contexts are arrows of type $1 \to 1$, while in logic programming arrows of type $\mathfrak{t}^n \to \mathfrak{t}^m$. Then the existence of RPOs is not needed for all the arrows of the base category, but only for such *contexts*.

Proposition 3. *In a reactive system with redex and context RPOs, where all contexts are reactive and ϕ defines a composition-reflecting subcategory, $\simeq_I^\phi$ is a congruence.*

As for bisimulation, we can define a semi-saturated version of ϕ-trace equivalence.

Definition 13. Let $\mathcal{R}$ be a reactive system, and ϕ a property on the arrows of **C**. We say that p and q are *semi-saturated ϕ-trace* equivalent ($p \simeq_{SS}^\phi q$) if the following holds:

- $\phi(p)$ if and only if $\phi(q)$,
- if $p \stackrel{l}{\twoheadrightarrow_I} p' \wedge \phi(p')$ then $q \stackrel{l}{\twoheadrightarrow_S} q'$ and $\phi(q')$,
- if $q \stackrel{l}{\twoheadrightarrow_I} q' \wedge \phi(q')$ then $p \stackrel{l}{\twoheadrightarrow_S} p'$ and $\phi(p')$,

where $\twoheadrightarrow_I$ and $\twoheadrightarrow_S$ are the transitive closures of $\to_I$ and $\to_{SAT}$.

As semi-saturated bisimilarity corresponds to saturated bisimilarity, semi-saturated ϕ-trace equivalence is saturated ϕ-trace equivalence, under the weak constraint of the existence of redex IPOs.

Theorem 4. *In a reactive system with redex IPOs, where all contexts are reactive, and such that ϕ defines a composition-reflecting subcategory, then $\simeq_{SS}^\phi = \simeq_{SAT}^\phi$.*

5. Logic programming

Logic programming, together with open input Petri nets (Example 4), points out that IPO abstract semantics are sometimes *too strict*, while saturated ones are, at some extent, more suitable. In this section, it turns out that saturated trace equivalence coincides with the ordinary logic semantics of logic programming, while IPO trace equivalence yields a finer semantics, known in the logic programming community as *S*-semantics [22].

A *logic signature* Γ is a pair (Σ, Π), where Σ is a set of *function symbols* and Π is a set of *predicate symbols* with an associated arity. As usual, given a set X of variables, we denote by $T_\Sigma(X)$ the free Σ-algebra over X. A *term* over X is an element of $T_\Sigma(X)$. Given a term t, $\mathrm{Var}(t)$ is the smallest set of names X such that $t \in T_\Sigma(X)$. An *atomic formula* over X has the form $P(t_1, \ldots, t_n)$ where P is a predicate with arity n, and $t_1, \ldots, t_n$ are terms over X. A *formula* is a finite conjunction of atomic formulas: $a_1 \wedge \cdots \wedge a_n$ where $\wedge$ is associative and it has the *empty formula* $\Box$ as unit. Note that in the standard definition $\wedge$ is also commutative, but to simplify our construction, as it is the case in Prolog, we do not consider it to be commutative (however the resulting behavior is the same).

If X and Y are sets of variables, a *substitution* from X to Y is a function $\sigma : X \to T_\Sigma(Y)$. If t is a term over X and σ a substitution from X to Y, then the term over Y, obtained by simultaneously substituting in t all the occurrences of the variables in X with their image under σ, is called the application of σ to t and written $t; \sigma$ (or $\sigma(t)$). If σ is a substitution from X to Y, and σ' from Y to Z, then $\sigma; \sigma'$ from X to Z is defined by applying σ' to each image of the variables in X under σ. Given $\sigma : X \to T_\Sigma(Y)$ and $X' \subseteq X$ the *restriction* of σ to X', written $\sigma \restriction X'$, is the substitution $\sigma' : X' \to T_\Sigma(Y)$ acting as σ on X'.

A substitution σ is *more general* than σ' if there exists a substitution θ such that $\sigma' = \sigma; \theta$. Two substitutions ψ and ϕ *unify* if there exists a substitution σ such that $\psi; \sigma = \phi; \sigma$, in this case σ is a *unifier* of ψ and ϕ. It is well-known that if ψ and ϕ unify, then there exists a unifier that is more general than all the others, called the *most general unifier* (*mgu* for short). It is also well-known that an *mgu* is the coequalizer in the category of substitutions [26], and in [15] it is shown that the *mgu* of substitutions with disjoint sets of variables corresponds to a pushout (this will be detailed later).

A *logic program* is a finite collection of *Horn clauses*, i.e., expressions of the form $h :- b$ where h is an atomic formula called the *head* of a clause, and b is a formula called the *body*. Rules in Table 1 define the operational semantics of logic programming. A goal $g = a_1 \wedge \cdots \wedge a_n$ reacts with a clause $c = h :- b$ if a_i, an atomic formula of the goal g, unifies with $\rho(h)$ (where ρ substitutes the variables of h with fresh variables not appearing in g). Let σ be the *mgu* of a_i and $\rho(h)$, then g reacts and becomes $g' = \sigma(a_1) \wedge \cdots \wedge \sigma(a_{i-1}) \wedge \sigma(b) \wedge \sigma(a_{i+1}) \wedge \cdots \wedge \sigma(a_n)$. A *refutation* of g is a derivation $g \Rightarrow_{\sigma_1} g_2 \Rightarrow_{\sigma_2} \cdots \Rightarrow_{\sigma_n} g_n$ ending with the empty formula (i.e. $g_n = \Box$). In this case $\sigma = \sigma_1; \ldots; \sigma_n \restriction \mathrm{Var}(g)$ is a *computed answer substitution* of g.

Table 1
Operational rules for SLD-resolution.

$$\frac{h :\!- b \in P \quad \sigma = mgu(a, \rho(h))}{P \Vdash a \Rightarrow_\sigma \sigma(\rho(b))} \quad \rho \text{ renames to globally fresh names}$$

$$\frac{P \Vdash g \Rightarrow_\sigma f}{P \Vdash g_1 \wedge g \wedge g_2 \Rightarrow_\sigma \sigma(g_1) \wedge f \wedge \sigma(g_2)}$$

5.1. Goals equivalences

Given a logic program, when are two goals equivalent? First note that we already have an LTS, but bisimulation is quite uninteresting in this case because we would like to consider as equivalent two goals with different branching behavior. Here the interesting point is if, and when, two goals can be refuted. The first naive equivalence that comes to mind is: g_1 can be refuted iff g_2 can be refuted. This equivalence is, however, very coarse and equates a lot of goals that we would like to distinguish.

Logic equivalence (denoted by $\simeq_L$) equates g_1 and g_2 if and only if, for any ground substitution σ, $\sigma(g_1)$ is refuted iff $\sigma(g_2)$ is refuted. In [22], *S-equivalence* (denoted by $\simeq_S$) is proposed: g_1 and g_2 have the same set of computed answer substitutions. Another interesting equivalence is *correct answer equivalence* (denoted by $\simeq_C$) that equates two goals iff they have the same set of correct answer substitutions (defined as follows). Let $\xrightarrow{\sigma}$ be the transition system defined by changing the premise of the first rule of Table 1: we do not require anymore that σ is the mgu, but only that it unifies a and $\rho(h)$ i.e, $\sigma(a) = \sigma(\rho(h))$. If $g \xrightarrow{\sigma_1} g_2 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} \Box$ we say that $\sigma = \sigma_1; \ldots; \sigma_n \restriction \mathrm{Var}(g)$ is a *correct answer substitution* of g. In other words σ is a correct answer substitution of g iff $\sigma(g)$ is a logical consequence of the program.

In [15], it is shown that, if we work with an infinite set of function symbols, $g_1 \simeq_L g_2$ iff $g_1 \simeq_C g_2$.

The following example shows that S-equivalence is somehow too detailed and that logic equivalence is more abstract.

Example 7. Consider the following program, where y is a variable and a is a constant:

$$P(y) :\!- \Box \qquad P(a) :\!- \Box \qquad Q(y) :\!- \Box.$$

Now consider the goals $P(x)$ and $Q(x)$. They are refuted by any ground substitution, which means that they are logic equivalent (and also correct answer equivalent). However, they are not S-equivalent: in fact the set of computed answer substitutions for $P(x)$ is $\{\epsilon, [a/x]\}$, while the computed answer substitutions for $Q(x)$ are $\{\epsilon\}$.

In Section 5.3, we will show that IPO trace equivalence coincides with S-equivalence and thus it is too strict since it distinguishes the goals $P(x)$ and $Q(x)$ defined above, while saturated trace equivalence exactly coincides with correct answer equivalence (and thus logic equivalence) and thus it cannot distinguish between $P(x)$ and $Q(x)$.

5.2. Logic programs as reactive systems

Here we show how logic programs can be seen as reactive systems. This will be used to prove, later, that saturated semantics correspond to logic equivalence, while standard semantics to the finer S-equivalence.

Consider two basic sorts t for terms and p for formulas (predicates are atomic formulas). We use ϵ to denote the empty string and t^n to denote the string composed of n occurrences of t. Given a logic signature $\Gamma = (\Sigma, \Pi)$, we define Γ' as the signature Γ enriched with the symbols $\wedge$ that takes two formulas and returns one formula and $\Box$ a constant formula. Let E be the set of axioms describing that $\wedge$ is associative (not commutative) and has identity $\Box$. Let X_p and X_t be sets of predicate and term variables. We use $T_{\Gamma'/E}(X_p, X_t)$ to denote the Γ'-algebra freely generated by (X_p, X_t) quotiented by E. A substitution for this algebraic specification is a function $\sigma : (X_p, X_t) \to T_{\Gamma'/E}(Y_p, Y_t)$. A term of this algebra in sort p is a logic formula having term and predicate variables from X_t and X_p.

In order to model logic programs as reactive systems, we will use the category $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$, that is the dual category of $\mathbf{Th}[\Gamma'/\mathbf{E}]$, i.e., the *Lawvere theory* [32] associated to the specification Γ', E. This category has been used in [15] as base category for a tile system of logic programming. Since the Lawvere theory is usually well-known for one sorted signature, here we are considering a two-sorted signature, and we report, below, the definition of $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$.

Definition 14. The category $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$ is defined as follows.

- objects are strings $s \in \{\mathsf{t}, \mathsf{p}\}^*$ representing ordered canonical variables,
- arrows $s_1 \to s_2$ are substitutions assigning to each variable in s_1 a term of $T_{\Gamma'/E}$ with variables in s_2,
- the identity arrow is the identity substitution and
- composition of arrows is composition of substitutions.

As an example of an object, consider the string $\mathsf{p}^n\mathsf{t}^m$. This represents n ordered *canonical predicate variables* (i.e., variables indexed from 1 to n) $p_1, \ldots, p_n$ and m ordered *canonical term variables* $x_1, \ldots, x_m$. To avoid confusion, it must be clear that the canonical variables are just placeholders, i.e., their scope is only local. For example, in $[f(x_1)/x_1]$ the two x_1 are different, while in $[f(x_1)/x_1, g(x_1)/x_2]$ only the two occurrences of x_1 in $f(x_1)$ and $g(x_1)$ refer to the same

placeholder. Note that in Definition 2 we talked about *holes*, while here we are talking about variables; moreover, here we are considering substitutions instead of tuples of terms. Notice that each tuple of terms can be regarded as a substitution of canonical variables and vice versa. For example, pt-tuple of terms $\langle P(f(x_1)), g(x_2)\rangle : \mathsf{pt} \to \mathsf{t}^n$ is the substitution $[P(f(x_1))/p_1, g(x_2)/x_1] : \mathsf{pt} \to \mathsf{t}^n$. We will often refer to the arrows of $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$ as both tuples of terms and substitutions.

Arrows of the form $\mathsf{t}^n \to \mathsf{t}^m$ are finite substitutions on Σ (with canonical sets of variables) and the arrows $\mathsf{t} \to \epsilon$ are closed terms over Σ, while arrows $\mathsf{p} \to \epsilon$ are closed formulas over Γ'. Arrows $\mathsf{p} \to \mathsf{t}^n$ are formulas over n canonical term variables, while arrows $\mathsf{p} \to \mathsf{pt}^n\mathsf{p}$ are formulas over n canonical term variables and two canonical predicate variables. Consider for example $\langle P(x_1, x_2) \wedge p_1, f(x_1), Q(f(x_2)), p_5\rangle$ where x_1, x_2 are terms variables and p_1, p_5 are predicate variables. This tuple corresponds to an arrow from ptp^2 to $\mathsf{t}^2\mathsf{p}^5$. Note also that the above tuple can represent also an arrow from ptp^2 to tptp^4.

Furthermore the above tuple can be seen as an arrow having as codomain objects $\mathsf{t}^n\mathsf{p}^m$ for $n \geq 2$ and $m \geq 5$, i.e., the codomain does not define the exact index of (term or predicate) variables, but the maximum index that the variables can have. In the following, for a goal g and a natural number n larger than the maximal index of variables appearing in g, we will write g^n to denote the arrow $\mathsf{p} \to \mathsf{t}^n$.

In the classical interpretation by Leifer and Milner, the arrows having domain objects different from 0 (the distinguished object) are seen as contexts which can be pre-composed with terms. In our reactive system, these arrows are substitutions which instantiate the variables of formulas. Horn clauses not only must be instantiated by substitutions, but they must be also contextualized with the $\wedge$ operator.

In the remainder of this section we will use

- the formula $f_1 = P(s(x_1), x_2) \wedge P(x_1, t(x_3))$ and
- the clause $c_1 = P(y_1, t(y_2)) :\!-\, Q(y_1)$

as a running example. The head of the c_1 must be instantiated (e.g., substituting y_1 with x_1 and y_2 with x_3) and contextualized (plugging it into $P(s(x_1), x_2) \wedge [-]$) in order to match f_1.

Similar problems arise with process calculi where the rules usually are not ground, and have to be instantiated and contextualized. For example, the left hand side of the CCS rule $a.P \mid \overline{a}.Q \rightsquigarrow P \mid Q$ matches $\nu a.(a.\mathbf{0} \mid \overline{a}.\mathbf{0})$ instantiating P, Q to $\mathbf{0}$ and plugging the left-hand side into the context $\nu a.[-]$. Usually this problem is avoided by creating infinitely many rules corresponding to all possible instantiations of the rule, and then considering only contextualization, as it is done for bigraphs [37]. This approach causes the problem of having infinitely many rules and consequently infinitely many transitions. For logic programming, we use an approach that is analogous to the one adopted for CCS in [8], i.e., we consider arrows that can both contextualize and instantiate. Here we simulate contextualization by substitutions by supplying appropriate variables in the rules. The redex of a rule is not simply an arrow of the form $h : \mathsf{p} \to \mathsf{t}^n$ that can only be instantiated, but it is an arrow $p_1 \wedge h \wedge p_2 : \mathsf{p} \to \mathsf{pt}^n\mathsf{p}$ that can be instantiated and contextualized (by instantiating the variables p_1 and p_2). In this way, we also get a finite branching IPOTS.

Thus, in our reactive system, the head of the clause c_1 above becomes $p_1 \wedge P(y_1, t(y_2)) \wedge p_2$ and, in this way, it can match the goal by instantiating p_1 to $P(s(x_1), x_2)$, p_2 to $\Box$ and y_1 to x_1 and y_2 to x_3.

Summarizing, we can say that we allow only substitutions and simulate contextualizations by substitutions by supplying appropriate variables in the rules (see below). In order to integrate this idea with the theory of reactive systems, we have "reversed" the arrows, i.e., a formula over n term variables becomes $\mathsf{p} \to \mathsf{t}^n$ (instead of the maybe more intuitive $\mathsf{t}^n \to \mathsf{p}$).

Definition 15. Given a logic program P on a signature Γ, we define a reactive system $\mathcal{R}(P)$ as follows:

1. $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$ is the underlying category,
2. p is the distinguished object,
3. all contexts are reactive,
4. for each clause $h :\!-\, b$, let n be the largest index of variables contained in h and b; then we add the rule

$$(p_1 \wedge h \wedge p_2 \ , \ p_1 \wedge b \wedge p_2)$$

where left and right-hand sides are arrows $\mathsf{p} \to \mathsf{pt}^n\mathsf{p}$ and p_1, p_2 are predicate variables.

Note that h and b do not necessarily have the same number of variables, while our theory requires that the left-hand and right-hand side of a rule have the same interface (i.e., they must be arrows with the same target). In this case, we extend the smaller interface.

Recall the definition of *redex square* (Section 2). A generic redex square for the above defined reactive system is depicted in diagram (i) of Fig. 3. Arrow c is a substitution that instantiates the variables of g, while arrow d instantiates the variables of h and contextualizes h, instantiating the predicate variables p_1 and p_2. Thus, for any reaction step, an atom of the goal is unified with the head of a clause and p_1 is instantiated with the formula on the left of the chosen atom, and p_2 is instantiated with the formula on the right.

Lemma 1. *The exterior square of diagram (i) in Fig. 3 commutes if and only if there exist formulas g_1, g_2 and an atomic formula a such that $g = g_1 \wedge a \wedge g_2$, p_1; $d = g_1$; c, p_2; $d = g_2$; c and h; $d = a$; c.*

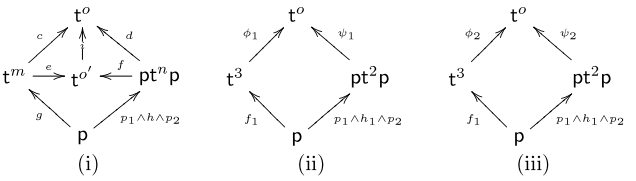


Fig. 3. (i) A generic redex square and a candidate for it. (ii,iii) Two redex squares for $f_1 = P(s(x_1), x_2) \wedge P(x_1, t(x_3))$ and h_1 the head of the clause $c_1 = P(y_1, t(y_2)) :- Q(y_1)$. The substitution ϕ_1 is $[x_1/x_1,\ t(x_2)/x_2,\ x_3/x_3]$, ψ_1 is $[\Box/p_1,\ P(x_1, t(x_3)),\ s(x_1)/y_1,\ x_2/y_2]$, ϕ_2 is $[x_1/x_1, x_2/x_2, x_3/x_3]$ and ψ_2 is $[P(s(x_1), x_2)/p_1,\ \Box/p_2,\ x_1/y_1,\ x_2/y_2]$.

Proof. Suppose that $g = g_1 \wedge a \wedge g_2$ and $p_1; d = g_1; c, h; d = a; c$ and $p_2; c = g_2; c$. Then $p_1 \wedge h \wedge p_2; d = g_1 \wedge a \wedge g_2; c$, i.e., the exterior square of diagram (i) in Fig. 3 commutes. For the other direction, recall that the operator $\wedge$ is associative and it has as an identity element. Thus, there always exists a, g_1 and g_2 such that $g = g_1 \wedge a \wedge g_2$. If the exterior square commutes, then $g_1 \wedge a \wedge g_2; c = p_1 \wedge h \wedge p_2; d$. Since $\wedge$ is not commutative, then $g_1; c = p_1; d, g_2; c = p_2; d$ and $a; c = h; d$. □

In general, in $\mathcal{R}(P)$, given a rule and a goal, there exist several ways of unifying them: one for each atom of the goal that can match the head h. Consider, for example, c_1 and f_1 described above. The head of c_1 unifies both with the left predicate of f_1 and with the right one, as illustrated by diagrams (ii) and (iii) in Fig. 3. This means that, given a rule and a goal – seen as arrows – there usually exists no minimal way of matching them (i.e., no pushout exists). The following lemma ensures that each commuting square fixes a "way" of matching, i.e., chooses the atom of the goal that unifies h.

Lemma 2. *Let the exterior square in diagram* (i) *of Fig. 3 be commuting. Let g_1, a, g_2 be formulas as described in Lemma 1. Then, for each candidate $\langle e, f, i\rangle$, the following hold: $p_1; f = g_1; e$, $p_2; f = g_2; e$ and $h; f = a; e$.*

Proof. Since the source of e is of type t^o, then also its target cannot contain predicate variables. Since the target of e is the same as the target of f, then f must instantiate the predicate variables p_1 and p_2 or, in other words, f must choose which atom of g matches the head h. Since $f; i = d, f$ is forced to make the same choice of d. Then, by Lemma 1, $p_1; f = g_1; e$, $p_2; f = g_2; e$ and $h; f = a; e$. □

As a next step, we are going to show that, in our reactive system, a redex RPO is the *mgu* of a and h, together with the instantiation of p_1 and p_2 to appropriate formulas. We start by recalling a theorem from [15].

Theorem 5. *The pushout between the arrows $a : \mathrm{t}^o \to \mathrm{t}^m$ and $b : \mathrm{t}^o \to \mathrm{t}^n$ is the most general unifier between a and $\rho(b)$ where ρ renames the variables $\{x_1 \ldots x_n\}$ to a globally fresh name.*

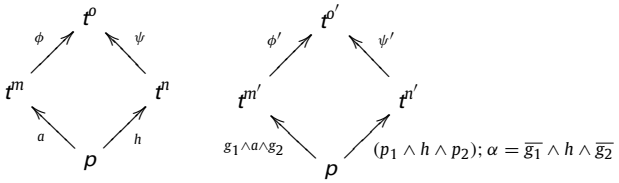
Proof. First of all, notice that the theorem states the correspondence only for the arrows of type $\mathrm{t}^n \to \mathrm{t}^m$ rather than all the arrows of $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$. It is easy to see that the full subcategory of $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$ containing objects only of type t^n is isomorphic to $\mathbf{Th}[\Sigma]^{op}$. It is well known from [26] that the mgu of two substitutions is the coequalizer of the corresponding arrows in $\mathbf{Th}[\Sigma]^{op}$. However, in Logic Programming, we are interested only in unification between substitutions whose set of variables are disjoint, since the variables in the head of the selected clause have been renamed on purpose to be different from those in the selected goal. This observation has been first made in [15], where it is shown that the pullback in $\mathbf{Th}[\Sigma]$ coincides with the mgu of these "renamed substitutions". Our theorem follows by the aforementioned result, by duality (i.e., pullbacks in $\mathbf{Th}[\Sigma]$ are pushouts in $\mathbf{Th}[\Sigma]^{op}$).

In order to give a stronger intuition of the correspondence between mgu and pushout, we show an example. Consider $a = [f(x_1, x_2)/x_1]$ and $b = [x_1/x_1]$. These are arrows $a : \mathrm{t}^1 \to \mathrm{t}^2$, $b : \mathrm{t}^1 \to \mathrm{t}^1$. The pushout in $\mathbf{Th}[\Sigma]^{op}$ (and thus in $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$) of these arrows is a pair of arrow $\langle \phi : \mathrm{t}^2 \to \mathrm{t}^2, \psi : \mathrm{t}^1 \to \mathrm{t}^2\rangle$, where $\phi = [x_1/x_1, x_2/x_2]$ and $\psi = [f(x_1, x_2)/x_1]$. The most general unifier between a and $\rho(b) = [x_3/x_1]$ is a substitution $[x_1/x_1, x_2/x_2, f(x_1, x_2)/x_3]$. □

Remember that if two substitutions can unify, then there exists an *mgu*. This, together with Theorem 5, ensures that, for each commuting square of substitutions, there exists a pushout. Moreover, this result holds not only for substitutions but also for atomic goals, since two atomic goals unify iff they consist of the same predicate and the terms within the predicate unify. In the remainder of this section we use $\overline{g}$ to denote a formula having the same predicate symbols as g, but without function symbols and where all variables are different. For example $\overline{f_1} = P(u_1, u_2) \wedge P(u_3, u_4)$. Note that the arrow d of a generic redex square (see diagram (i) in Fig. 3) can always be decomposed into $\alpha; \psi'$ where α instantiates p_1 and p_2 to $\overline{g_1}$ and $\overline{g_2}$ and ψ' is a substitution. It is exactly this arrow α that chooses which atom of the goal matches h.

The following lemma generalizes the theorem above to non-atomic formulas of the form $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge b \wedge \overline{g_2}$.

Lemma 3. *Let a and h be atomic formulas. Then $\langle \phi, \psi\rangle$ is the pushout of a and h (depicted in diagram* (i) *below) if and only if $\langle \phi', \psi'\rangle$ is the pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$ (diagram* (ii)*), where ϕ' is equal to ϕ on $\mathrm{Var}(a)$ and the identity on the others variables, and ψ' is equal to ψ on $\mathrm{Var}(h)$ and such that $g_1; \phi' = \overline{g_1}; \psi'$ and $g_2; \phi' = \overline{g_2}; \psi'$.*



(i) (ii)

Proof. Suppose that diagram (i) is a pushout, and suppose that there exists u', v' such that $g_1 \wedge a \wedge g_2; u' = \overline{g_1} \wedge h \wedge \overline{g_2}; v'$. Then we have that $a; u = h; v$ for u, v, the restrictions of u' and v' on $\mathrm{Var}(a)$ and $\mathrm{Var}(h)$, respectively. Since diagram (i) is a pushout, then there exists a unique ρ such that $\phi; \rho = u$ and $\psi; \rho = v$.

Now, we can construct a ρ' such that $\phi'; \rho' = u'$ and $\psi'; \rho' = v'$. Take $\rho' = \rho$ on $\mathrm{Var}(a; \phi)$ and equal to u' on all the other variables. Now, it is easy to see that $\phi'; \rho' = u'$. Indeed, for the variables in $\mathrm{Var}(a)$, $\phi' = \phi$, $\rho' = \rho$ and $u' = u$ and $\phi; \rho = u$. For the other variables ϕ' is, by construction, the identity and $\rho' = u'$. Now, we have to prove that $\psi'; \rho' = v'$. On $\mathrm{Var}(h)$, $\psi' = \psi$, $\rho' = \rho$ (observe that $\mathrm{Var}(a; \phi) = \mathrm{Var}(h; \psi)$) and $v' = v$ and $\psi; \rho = v$. For the other variables, recall that by construction $\overline{g_1}; \psi' = g_1; \phi'$ and $\overline{g_2}; \psi' = g_2; \phi'$. Then $\overline{g_1}; \psi'; \rho' = g_1; \phi'; \rho' = g_1; u' = \overline{g_1}; v'$ and $\overline{g_2}; \psi'; \rho' = g_2; \phi'; \rho' = g_2; u' = \overline{g_2}; v'$.

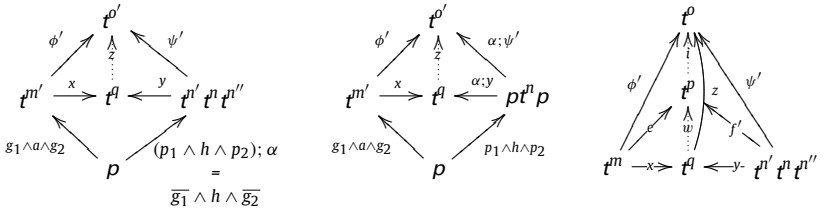
Now suppose that there exists another ρ'_1 such that $\phi'; \rho'_1 = u'$ and $\psi'; \rho'_1 = v'$. First of all notice that on $\mathrm{Var}(a; \phi')$, ρ'_1 must be equal to ρ', otherwise, we can construct a ρ_1 (different from ρ) such that $\phi; \rho_1 = u$ and $\psi; \rho_1 = v$. For all the other variables, notice that ϕ' is the identity, and thus in order to have $\phi'; \rho'_1 = u'$, ρ'_1 must be equal to u', i.e., equal to ρ'.

Now we prove the other direction. Suppose that diagram (ii) is a pushout and suppose that there exist u, v such that $a; u = h; v$. Let u' equal to u on $\mathrm{Var}(a)$ and the identity on the others. Let v' equal to v on $\mathrm{Var}(h)$ and such that $g_1; u' = \overline{g_1}; v'$ and $g_2; u' = \overline{g_2}; v'$. Then, $(g_1 \wedge a \wedge g_2); u' = \overline{g_1} \wedge h \wedge \overline{g_2}; v'$, because $g_1; u' = \overline{g_1}; v'$, $g_2; u' = \overline{g_2}; v'$ and $a; u' = a; u = h; v = h; v'$. Since, by hypothesis diagram (ii) is a pushout, then there exists a unique ρ' such that $\phi'; \rho' = u'$ and $\psi'; \rho' = v'$. Now we have that $\phi; \rho = u$ and $\psi; \rho = v$, for ρ equal to ρ' on $\mathrm{Var}(a; \phi)$. Now suppose that there exists a different ρ_1 such that that $\phi; \rho_1 = u$ and $\psi; \rho_1 = v$. Then, as proved in the other direction, we can construct ρ'_1 different from ρ' such that $\phi'; \rho'_1 = u'$ and $\psi'; \rho'_1 = v'$, but this is impossible, since ρ' is unique. □

The meaning of this lemma is more intuitive if one considers formulas. Suppose that a and h unify, and let $\langle \phi, \psi \rangle$ be their *mgu*. Then also $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$ unify and the *mgu* is the *mgu* of a and h (since all the variables of $\overline{g_1}$ and $\overline{g_2}$ are different and can be instantiated to $g_1; \phi$ and $g_2; \psi$).

The following lemma is central since it shows the relationship between RPOs and pushouts: if we fix a way of matching (the arrow α), then we have only one minimal unifier (i.e, pushout) while if we do not fix it, we have several minimal unifiers (i.e., RPOs) one for each way of matching (i.e., for each α).

Lemma 4. *Let a and h be atomic formulas, and α as described above i.e., such that $(p_1 \wedge h \wedge p_2); \alpha = \overline{g_1} \wedge h \wedge \overline{g_2}$. Suppose that the exterior square of diagram* (ii) *below commutes. Then $\langle x, y \rangle$ is the pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$, and z the mediating morphism (diagram* (i)*) iff $\langle x, \alpha; y, z \rangle$ is the RPO of the exterior square of diagram* (ii).



(i) (ii) (iii)

Proof. Let us consider the diagrams above. We suppose that $\langle x, y \rangle$ is the pushout and we prove that $\langle x, \alpha; y, z \rangle$ is the RPO. Let $\langle e, f, i \rangle$ be a candidate for the exterior square of diagram (ii). Thus, by Lemma 2, $e(g_1) = f(p_1)$, $e(g_2) = f(p_2)$ and $e(a) = f(h)$. This means that f factors through α, i.e., $f = \alpha; f'$, for f' such that $e(g_1) = f'(\overline{g_1})$, $e(g_2) = f'(\overline{g_2})$ and $e(a) = f'(h)$.

Since $\langle x, y \rangle$ is the pushout there exists a unique w such that $x; w = e$ and $y; w = f'$. From the latter, we have that $\alpha; y; w = \alpha; f' = f$. Now we have to prove that $w; i = z$, but this is trivial since z is the unique morphism such that $x; z = \phi'$ and $y; z = \psi'$.

Now we prove the other direction. Suppose that $\langle x, \alpha; y, z \rangle$ is the RPO of the diagram. Since $\langle x, \alpha; y, z \rangle$ is a candidate, by Lemma 2, we have that $x(a) = \alpha; y(h)$. As recalled after Theorem 5, if two atomic goals unifies, then there exists their most general unifier. Now, since a and h unifies then there exists the pushout of a and h (i.e., the mgu). Then, by Lemma 3 there exists $\langle e, f' \rangle$ as pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$. Let u be the unique mediating morphism such that $e; u = x$ and $f'; u = y$. Then $\langle e, \alpha; f', u \rangle$ is a candidate for the square $\langle g_1 \wedge a \wedge g_2, x, p_1 \wedge h \wedge p_2 \rangle, \alpha; y$. Notice that by Proposition 1 of [33], the latter square is an IPO. Thus there exists a unique w such that $x; w = e, \alpha; y; w = \alpha; f'$ and $w; u = id$. Now, recall that α only instantiates the predicate variables p_1 and p_2 with $\overline{g_1}$ and $\overline{g_2}$ that have globally new variables. Then, α is epi and thus $y; w = f'$. Now it remains to prove that $u; w = id$, but this is trivial. Indeed, since $\langle e, f' \rangle$ is a pushout, there exists a unique arrow v such that $e; v = e$ and $f'; v = f'$ and both id and u; w satisfy this requirement. □

Then, given a commuting square, this fixes a way of matching (i.e., one α) and so there exists a minimal unifier, that is the *mgu* between the head of a clause h and chosen atom a of the formula g.

Theorem 6. *$\mathcal{R}(P)$ has redex and context RPOs.*

Proof. Given a redex square as the one in Fig. 3(i), by Lemma 1 it identifies one atom of the goal that matches h, and the formulas at the left and at the right of the atom (a, g_1 and g_2). Recall that if two terms unify, then their *mgu* exists. Since a and h unify, their *mgu*, i.e., their pushout, exists. We call it $\langle \phi, \psi \rangle$. By Lemma 3, $\langle \phi', \psi' \rangle$ is the pushout between $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge a \wedge \overline{g_2}$ will exist. Now we can compose α with ψ' and we get, by Lemma 4, the RPO of the diagram.

Now we show that RPOs exist also for context squares. First of all note that in context squares all the arrows have the form $\mathrm{t}^m \to \mathrm{t}^n$. These are simple term substitutions and thus, if they commute (unify), then there exists a *mgu* (i.e. a pushout) of them, and it is for sure an RPO. □

5.3. Saturated and IPO abstract semantics

In this section, we show that S-equivalence corresponds to IPO ϕ-trace equivalence, while correct answer equivalence corresponds to saturated ϕ-trace equivalence (both of them are defined in Section 4.2).

Recall the definition of saturated (Definition 4) and IPO (Definition 8) transition system. In the former, a state f can perform a transition labeled with a context c going in the states g (in symbols $f \xrightarrow{c}_{SAT} g$) if and only if $c(f) \rightsquigarrow g$. This corresponds to $\longrightarrow$ (as defined in Section 5.1) where a formula f can perform a transition labeled with the substitution σ whenever σ unifies f with a redex. In the latter transition system, f can perform a transition labeled c (in symbols $f \xrightarrow{c}_{I} g$) only if c is the *minimal* context that allows $c(f) \rightsquigarrow g$. This minimal context is the smallest substitution that unifies the formula with the head of a clause (i.e., the *most general unifier*) and thus $\longrightarrow_I$ corresponds to $\Rightarrow$ (i.e., SLD transitions).

Theorem 7. *Let P be a logic program and $\mathcal{R}(P)$ the corresponding reactive system. Let f, g be two formulas and m, n larger than the maximal index of variables appearing in f and g. Furthermore let σ be a substitution, and let $\theta : \mathrm{t}^m \to \mathrm{t}^n$ be equal to σ on $\mathrm{Var}(f)$ and id otherwise. Then:*

- $P \Vdash g \xrightarrow{\sigma} f$ *iff in* $\mathcal{R}(P)$ *it holds that* $g^m \xrightarrow{\theta}_{SAT} f^n$,
- $P \Vdash g \Rightarrow_\sigma f$ *iff in* $\mathcal{R}(P)$ *it holds that* $g^m \xrightarrow{\theta}_{I} f^n$.

Proof. First, note that $P \Vdash g \Rightarrow_\sigma g'$ iff there exists $(h :- b) \in P$ and formulas a, g_1, g_2 such that $g = g_1 \wedge a \wedge g_2$, $\sigma = mgu(a, \rho(h))$ and $g' = \sigma(g_1) \wedge \sigma(\rho(b)) \wedge \sigma(g_2)$.

Let c be equal to $\sigma \restriction \mathrm{Var}(a)$ and $d = \sigma \restriction \mathrm{Var}(h)$. By Theorem 5 $\langle c, d \rangle$ is the pushout of h and a, and by Lemmas 3 and 4, $\langle g, p_1 \wedge h \wedge p_2, c', d' \rangle$ is an IPO, where $c' \restriction \mathrm{Var}(a) = c$ and $c' = id$ on the others variables and $d' \restriction \mathrm{Var}(h) = d$ and it maps p_1, p_2 to $g_1; c'$, $g_2; c'$. Now, by construction, in $\mathcal{R}(P)$ there is a rule $p_1 \wedge h \wedge p_2 \to p_1 \wedge b \wedge p_2$, and then $g \xrightarrow{c'}_I (p_1 \wedge b \wedge p_2)$; $d' = c'(g_1) \wedge d(b) \wedge c'(g_2) = \sigma(g_1) \wedge \sigma(b) \wedge \sigma(g_2)$. For the other direction, we can proceed as before, by applying Lemma 4, Lemma 3 and Theorem 5.

For the other point, note that $P \Vdash g \xrightarrow{\sigma} g'$ iff there exists $(h : -b) \in P$ and formulas a, g_1, g_2 such that $g = g_1 \wedge a \wedge g_2$ and $\sigma(a) = \sigma(\rho(h))$ and $g' = \sigma(g_1) \wedge \sigma(\rho(b)) \wedge \sigma(g_2)$.

Now we can proceed as before without thinking back to mgu or IPO redex square, but only to unifiers and redex square. □

Corollary 1. *In $\mathcal{R}(P)$ the* IPOTS *is finite-branching.*

Note that S-equivalence and correct answer equivalence are ϕ-trace equivalence (Definition 12) where the predicate ϕ holds only for the empty goal. Formally we define the predicate $\Box()$ over all the arrows of the category $\mathbf{Th}[\Gamma'/\mathbf{E}]^{op}$: $\Box(a)$ holds iff a is an arrow obtained by decomposing $\Box^n : \mathrm{p} \to \mathrm{t}^n$, where $\Box^n$ is $\Box : \mathrm{p} \to \epsilon$ with the interface extended with n extra term variables. Essentially $\Box()$ holds for all term substitutions and for empty formulas. The predicate $\Box()$ defines a composition reflecting subcategory and, since all contexts are reactive, we can apply our theoretical results (Proposition 2, Proposition 3 and Theorem 4) to $\simeq^{\Box}_{I}$, $\simeq^{\Box}_{SAT}$ and $\simeq^{\Box}_{SS}$: these three equivalences are congruences (w.r.t. substitutions) and $\simeq^{\Box}_{SAT} = \simeq^{\Box}_{SS}$.

Now we show that the first corresponds to $\simeq_S$, while the second (and then also the third) correspond to $\simeq_C$ (that, in the case of infinitely many function symbols, is $\simeq_L$).

Theorem 8. *Let P be a logic program and $\mathcal{R}(P)$ be the corresponding reactive system. Then $\simeq_S = \simeq^{\Box}_{I}$ and $\simeq_C = \simeq^{\Box}_{SAT}$.*

Proof. Suppose that $p \simeq_S q$ and $p \xrightarrow{\theta}_I \Box$. Then $\exists\ \theta_1, \ldots, \theta_n$ such that $\theta_1; \theta_2; \ldots; \theta_n = \theta$ and $p \xrightarrow{\theta_1}_I p_2 \cdots \xrightarrow{\theta_n}_I \Box$. By Theorem 7 we have that $p \Rightarrow_{\sigma_1} p_2 \ldots p_n \Rightarrow_{\sigma_n} p'$ with $\theta_i = \sigma_i \restriction \mathrm{Var}(p_i)$. Note that $\sigma_i \restriction \mathrm{Var}(p_i); \sigma_{i+1} \restriction \mathrm{Var}(p_{i+1})$ is equal to $(\sigma_i; \sigma_{i+1}) \restriction \mathrm{Var}(p_i)$, and $\theta = (\sigma_1; \ldots \sigma_n) \restriction \mathrm{Var}(p_1)$. Since θ is a computed answer substitution of p and $p \simeq_S q$, θ is a computed answer substitution of q and $q \Rightarrow_{\phi_1} q_2 \Rightarrow_{\phi_2} q_3 \cdots \Rightarrow_{\phi_m} \Box$ such that $\phi_1; \phi_2; \ldots; \phi_m \restriction \mathrm{Var}(q) = \theta$. By Theorem 7 $q \xrightarrow{\psi_1}_I q_2 \xrightarrow{\psi_2}_I q_3 \cdots \xrightarrow{\psi_m}_I \Box$ where $\psi_i = \phi_i \restriction \mathrm{Var}(q_i)$. As before $\psi_1; \psi_2; \ldots; \psi_m = \phi_1 \restriction \mathrm{Var}(q_1); \phi_2 \restriction \mathrm{Var}(q_2); \ldots; \phi_m \restriction \mathrm{Var}(q_m) = \phi_1; \phi_2; \ldots; \phi_m \restriction \mathrm{Var}(q) = \theta$. Hence $q \xrightarrow{\theta}_I \Box$. The other direction is analogous.

To prove the second equivalence we use Theorem 7 and we can proceed as before. □

6. Coalgebras (on presheaves)

In this section we first introduce the basic notions of the theory of coalgebras [47] and then coalgebras on presheaves [53,24,23].

Definition 16 (*Category of Coalgebras and Cohomomorphisms*). Let $\mathbf{B} : \mathbf{C} \to \mathbf{C}$ be an endofunctor on a category $\mathbf{C}$. A *coalgebra for* $\mathbf{B}$ or $\mathbf{B}$-*coalgebra* is a pair $\langle X, \alpha \rangle$ where X is an object of $\mathbf{C}$ (called the *carrier*) and $\alpha : X \to \mathbf{B}(X)$ is an arrow (called the *transition structure*). A $\mathbf{B}$-*cohomomorphism* $h : \langle X, \alpha \rangle \to \langle Y, \beta \rangle$ is an arrow $h : X \to Y$ of $\mathbf{C}$ such that $\alpha; \mathbf{B}(h) = h; \beta$.

$\mathbf{B}$-coalgebras and $\mathbf{B}$-cohomomorphisms form the category $\mathbf{Coalg_B}$.

Different kinds of dynamical systems (e.g. deterministic automata, Moore and Mealy machines) can be seen as coalgebras by choosing a certain endofunctor on the category **Set**. For example, transition systems that are labeled over a set of label L are in one-to-one correspondence with $\mathbf{P_L}$-coalgebras [47], for $\mathbf{P_L}$ defined below.

Definition 17. Let L be a fixed set of labels and $\mathbf{P}$ be the powerset functor. The functor $\mathbf{P_L} : \mathbf{Set} \to \mathbf{Set}$ is defined as follows. For each set X, $\mathbf{P_L}(X) = \mathbf{P}(L \times X)$. For each function h, $\mathbf{P_L}(h) = \mathbf{P}(L \times h)$.

One desirable property of a category of coalgebras $\mathbf{Coalg_B}$ is *the existence of a final coalgebra* $1_\mathbf{B}$, that is a $\mathbf{B}$-coalgebra such that for any other $\mathbf{B}$-coalgebras $\langle X, \alpha \rangle$ there exists a unique cohomomorphism $!^{\mathbf{B}}_{\langle X,\alpha\rangle} : \langle X, \alpha \rangle \to 1_\mathbf{B}$. Indeed, if a final coalgebra exists then two elements of the carrier of a coalgebra are bisimilar if and only if they are mapped into the same element by the final cohomomorphism. This is theoretically very important, because it allows us to define the abstract semantics as a function (i.e., the unique morphism to a final coalgebra) that maps each system into the canonical representative of its equivalence class (i.e., its image through the final morphism).

Unfortunately, due to cardinality reasons, the category of $\mathbf{P_L}$-coalgebras does not have a final object [47]. One satisfactory solution consists in replacing the powerset functor $\mathbf{P}$ by the *countable* powerset functor $\mathbf{P}_c$, which maps a set to the family of its countable subsets. Then, by defining the functor $\mathbf{P}^\mathbf{c}_\mathbf{L} : \mathbf{Set} \to \mathbf{Set}$ as $X \mapsto \mathbf{P}_c(L \times X)$, one has that coalgebras for this endofunctor are one-to-one with transition systems with *countable branching degree*. Unlike functor $\mathbf{P_L}$, functor $\mathbf{P}^\mathbf{c}_\mathbf{L}$ admits final coalgebras (Example 6.8 of [47]).

When considering *nominal calculi*, i.e., those able to generate and communicate names, labeled transition systems are often too rough. For this reason, *indexed labeled transition systems* have been introduced in [17]. The coalgebraic models for this kind of systems consist of colagebras not on the category **Set**, but on some category of presheaves $\mathbf{Set^C}$, for some index category $\mathbf{C}$. For example, the early and late semantics of the π-calculus [41] can be characterized by proper endo-functors on $\mathbf{Set^I}$ [53,23,24] where $\mathbf{I}$ is the category of finite sets of names and injective functions. Moreover, by choosing different index categories, we can also characterize the open semantics of π-calculus [48] (via an endo-functor on $\mathbf{Set^D}$ [25,36]) and the semantics of explicit fusion calculus [55] (via an endo-functor on $\mathbf{Set^E}$ [7]).

Objects of $\mathbf{Set^C}$ are presheaves on $\mathbf{C}$, i.e., functors from $\mathbf{C}$ to **Set**. Intuitively, $\mathbf{X} \in |\mathbf{Set^C}|$ maps each object $i \in |\mathbf{C}|$ into a set, that represents the set of systems having i as *interface* and it maps each arrow $c \in \mathbf{C}[i, j]$ into a function that transforms systems with interface i into systems with interface j.

Arrows of $\mathbf{Set^C}$ are *natural transformations* between presheaves. Given two presheaves $\mathbf{X}$ and $\mathbf{Y}$, a natural transformation $h : \mathbf{X} \Rightarrow \mathbf{Y}$ is a $|\mathbf{C}|$-indexed family of functions $h = \{h_i : \mathbf{X}(i) \to \mathbf{Y}(i) \mid i \in |\mathbf{C}|\}$ such that for all $c \in \mathbf{C}[i, j]$ $h_i; \mathbf{Y}(c) = \mathbf{X}(c); h_j$.

From an algebraic perspective, one can think of presheaves as algebras of a multi-sorted unary specification where sorts and unary operators are, respectively, objects and arrows of $\mathbf{C}$. In this perspective, natural transformations are simply homomorphisms between algebras. Thus, if there exists a final coalgebras in $\mathbf{Coalg_B}$ (for some $\mathbf{B} : \mathbf{Set^C} \to \mathbf{Set^C}$), then the final morphism is a natural transformation (an homomorphism that respects the arrows of $\mathbf{C}$) and thus, bisimilarity is a congruence with respect to all the arrows of $\mathbf{C}$.

Proposition 4. *Let* $\mathbf{C}$ *be a category and* $\mathbf{B} : \mathbf{Set^C} \to \mathbf{Set^C}$ *be an endofunctor. Let* $\langle \mathbf{X}, \alpha \rangle$ *be a* $\mathbf{B}$*-coalgebra. If* $\mathbf{Coalg_B}$ *has a final coalgebra, then bisimilarity is a congruence with respect to the arrows in* $||\mathbf{C}||$*, i.e.,* $\forall i \in |\mathbf{C}|$, $\forall x, y \in \mathbf{X}(i)$ *if* $!^{\mathbf{B}}_{\langle \mathbf{X},\alpha\rangle}(x) = !^{\mathbf{B}}_{\langle \mathbf{X},\alpha\rangle}(y)$ *then* $\forall j \in |\mathbf{C}|$, $\forall c \in \mathbf{C}[i, j]$, $!^{\mathbf{B}}_{\langle \mathbf{X},\alpha\rangle}(\mathbf{X}(c)(x)) = !^{\mathbf{B}}_{\langle \mathbf{X},\alpha\rangle}(\mathbf{X}(c)(y))$.

7. Reactive systems as coalgebras on presheaves

In this section, we provide coalgebras over presheaves corresponding to the reaction relation, the saturated and the IPO transition systems for a certain reactive system $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$. We will provide these models at two different levels. In Section 7.1, we will provide coalgebras for endofunctors over $\mathbf{Set}^{|\mathbf{C}|}$, i.e., the category of presheaves over the category $|\mathbf{C}|$ (that contains only objects) and then, in Section 7.2, for endofunctors on $\mathbf{Set^C}$. At the first level, bisimilarity is not guaranteed to be a congruence (since there are not arrows in the category $|\mathbf{C}|$), while at the second level, this is ensured by Proposition 4.

Hereafter, we always assume that the base category $\mathbf{C}$ is small, i.e., $|\mathbf{C}|$ and $||\mathbf{C}||$ are proper sets (and not classes). Moreover we assume that $||\mathbf{C}||$ is a countable set. The latter requirement is needed to ensure that the reaction relation, the saturated and the IPO transition systems have countable branching degree. All the proofs are in Appendix B.

7.1. Coalgebraic models of reactive systems

Consider the covariant Yoneda embedding $\mathbf{y_0} : \mathbf{C} \to \mathbf{Set}$ (also denoted by $\mathbf{C}[0, -]$) where 0 is the distinguished object of $\mathcal{R}$. This presheaf associates to any object $i \in |\mathbf{C}|$ the set $\mathbf{C}[0, i]$, i.e., the set of arrows of $\mathbf{C}$ having source 0 and target i, while to any arrow $c \in \mathbf{C}[i, j]$ associates the function $-; c : \mathbf{C}[0, i] \to \mathbf{C}[0, j]$ that maps each arrow $p \in \mathbf{C}[0, i]$ into $p; c \in \mathbf{C}[0, j]$.

We will use this presheaf as the carrier of our coalgebraic models in the next subsection (where we will define models on $\mathbf{Set}^{\mathbf{C}}$). In this section, we will provide models for certain endofunctors on $\mathbf{Set}^{|\mathbf{C}|}$, and thus we will use $|\mathbf{y_0}| : |\mathbf{C}| \to \mathbf{Set}$, i.e., the restriction of $\mathbf{y_0}$ to the objects of $\mathbf{C}$. It is worth noting that a presheaf on $|\mathbf{C}|$ is just a $|\mathbf{C}|$-sorted set and a natural transformation $h : \mathbf{X} \Rightarrow \mathbf{Y}$ is simply a $|\mathbf{C}|$-sorted family of functions $\{h_i : \mathbf{X}(i) \to \mathbf{Y}(i) \mid i \in |\mathbf{C}|\}$.

Endofunctors on presheaves are usually defined as functors on $\mathbf{Set}$ acting pointwise. For example, the following endofunctor is simply the powerset functor defined pointwise. We choose to give an explicit definition, in order to better explain those endofunctors that will be introduced in the next subsection.

Definition 18. The endofunctor $\mathbf{R} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ is defined as follows.
For all objects $\mathbf{X} \in |\mathbf{Set}^{|\mathbf{C}|}|$, for all $i \in |\mathbf{C}|$, $[\mathbf{R}(\mathbf{X})](i) = \mathbf{P}_c(\mathbf{X}(i))$.
For all arrows $h \in ||\mathbf{Set}^{|\mathbf{C}|}||$, $\mathbf{R}(h) = \{\mathbf{P}_c(h_i) \mid i \in |\mathbf{C}|\}$.

An $\mathbf{R}$-coalgebra consists of a $|\mathbf{C}|$-sorted set $\mathbf{X}$ (i.e., $\mathbf{X} : |\mathbf{C}| \to \mathbf{Set}$) and a $|\mathbf{C}|$-sorted family of functions $\alpha = \{\alpha_i : \mathbf{X}(i) \to \mathbf{P}_c(\mathbf{X}(i)) \mid i \in |\mathbf{C}|\}$. Intuitively, α_i associates to any state with interface i, the set of states (all with interface i) that are reachable through an unlabeled transition. Thus, for each reactive systems $\mathcal{R}$, we can define the corresponding $\mathbf{R}$-coalgebra as $\langle |\mathbf{y_0}|, \alpha^{\mathcal{R}} \rangle$ where $\alpha_i^{\mathcal{R}}$ is defined $\forall p \in \mathbf{C}[0, i]$ as $\alpha_i^{\mathcal{R}}(p) = \{q \mid p \rightsquigarrow q\}$.

Definition 19. The endofunctor $\mathbf{D} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ is defined as follows.
For all objects $\mathbf{X} \in |\mathbf{Set}^{|\mathbf{C}|}|$,
for all $i \in |\mathbf{C}|$, $[\mathbf{D}(\mathbf{X})](i) = \mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i, j] \times \mathbf{X}(j))$.
For all arrows $h \in ||\mathbf{Set}^{|\mathbf{C}|}||$, $\mathbf{D}(h) = \{\mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i, j] \times h_j) \mid i \in |\mathbf{C}|\}$.

Thus, a $\mathbf{D}$-coalgebra consists of a $|\mathbf{C}|$-sorted set $\mathbf{X}$ and a $|\mathbf{C}|$-sorted family of functions $\alpha = \{\alpha_i : \mathbf{X}(i) \to \mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i, j] \times \mathbf{X}(j)) \mid i \in \mathbf{C}\}$. Now, α_i associates to any state with interface i, the set of its transitions where the label is an arrow in $\mathbf{C}[i, j]$ (for some j) and the arriving state has interface j. For each reactive systems $\mathcal{R}$, we can define the $\mathbf{D}$-coalgebra corresponding to the saturated and the IPO transition system as $\langle |\mathbf{y_0}|, \alpha_S^{\mathcal{R}} \rangle$ and $\langle |\mathbf{y_0}|, \alpha_I^{\mathcal{R}} \rangle$, respectively, where $\alpha_{S,i}^{\mathcal{R}}$ and $\alpha_{I,i}^{\mathcal{R}}$ are defined $\forall p \in \mathbf{C}[0, i]$ as $\alpha_{S,i}^{\mathcal{R}}(p) = \{(c, q) \mid p \xrightarrow{c}_{SAT} q\}$ and $\alpha_{I,i}^{\mathcal{R}}(p) = \{(c, q) \mid p \xrightarrow{c}_{I} q\}$.

7.2. Coalgebras on presheaves for reactive system

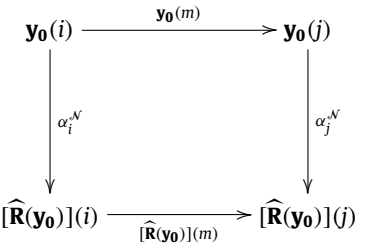
In the previous subsection we have provided models for reactive systems as coalgebras on $\mathbf{Set}^{|\mathbf{C}|}$. In this subsection, we lift these models to coalgebras on $\mathbf{Set}^{\mathbf{C}}$. These constructions provide both a characterization of $\sim^S$ and $\sim^I$ as final semantics and also an alternative proof of the fact that these are congruent with respect to the arrows in $\mathbf{C}$.

First of all, we would like to define an endofunctor $\widehat{\mathbf{R}} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ such that $\langle \mathbf{y_0}, \alpha^{\mathcal{R}} \rangle$ is an $\widehat{\mathbf{R}}$-coalgebra, but this is usually impossible since $\alpha^{\mathcal{R}}$ is not guaranteed to be an arrow of $\mathbf{Set}^{\mathbf{C}}$ (i.e., a natural transformation).

Example 8. Recall the reactive system $\mathcal{N} = \langle \mathbf{OPN}_N, 0, \mathbf{OPN}_N, \mathfrak{T} \rangle$ that has been introduced in Example 3 corresponding to the open input Petri net in Fig. 2(i). Recall that $\mathbf{OPN}_N$ has only two objects (0 and 1) and $\mathbf{OPN}_N[0, 0] = \mathbf{OPN}_N[0, 1] = \{a, b, c, d, e, f, x, y\}^{\oplus}$ and $\mathbf{OPN}_N[1, 1] = \{x, y\}^{\oplus}$. The presheaf $\mathbf{y_0} : \mathbf{OPN}_N \to \mathbf{Set}$ (i.e., $\mathbf{OPN}_N[0, -]$) associates to each object i, the set $\mathbf{OPN}_N[0, i]$ and to each arrow $m \in \mathbf{OPN}_N[i, j]$ the function $-; m : \mathbf{OPN}_N[0, i] \to \mathbf{OPN}_N[0, j]$ that coincides with $- \oplus m$, by definition of $\mathbf{OPN}_N$.

The $\mathbf{R}$-coalgebra (Definition 18) corresponding to $\mathcal{N}$ is defined as $\langle |\mathbf{y_0}|, \alpha^{\mathcal{N}} \rangle$ where $\alpha^{\mathcal{N}}$ maps each state p into the set of states q such that $p \rightsquigarrow q$. As an example, consider the multisets a, ax, c and cx of the open net in Fig. 2(i). We have that $cx \rightsquigarrow d$, while all the others cannot react. Thus the transition structure $\alpha^{\mathcal{N}}$ maps a, ax and c into $\emptyset$, while cx into d.

Now suppose that we would like to lift $\langle |\mathbf{y_0}|, \alpha^{\mathcal{N}} \rangle$ to a coalgebra over $\mathbf{Set}^{\mathbf{OPN}_N}$ for some behavioral endofunctor $\widehat{\mathbf{R}} : \mathbf{Set}^{\mathbf{OPN}_N} \to \mathbf{Set}^{\mathbf{OPN}_N}$. In order to have that $\langle \mathbf{y_0}, \alpha^{\mathcal{N}} \rangle$ is an $\widehat{\mathbf{R}}$-coalgebra, $\alpha^{\mathcal{N}}$ should be a an arrow in $\mathbf{Set}^{\mathbf{OPN}_N}[\mathbf{y_0}, \widehat{\mathbf{R}}(\mathbf{y_0})]$, that is, $\forall i, j \in |\mathbf{OPN}_N|$ and $\forall m \in \mathbf{OPN}_N[i, j]$ the following diagram should commute in $\mathbf{Set}$.



But, in our case, this is impossible for all functors $\widehat{\mathbf{R}}$. Indeed, take i and j as the object 1 and m as the multiset x. We have that $a, c \in \mathbf{OPN}_N[0,1]$ (that is $\mathbf{y_0}(1)$) and $\mathbf{y_0}(x)(a) = a$; $x = ax$ and $\mathbf{y_0}(x)(c) = c$; $x = cx$ and thus $\alpha_1^{\mathcal{N}}(\mathbf{y_0}(x)(a)) = \alpha_1^{\mathcal{N}}(ax) = \emptyset$ and $\alpha_1^{\mathcal{N}}(\mathbf{y_0}(x)(c)) = \alpha_1^{\mathcal{N}}(cx) = \{d\}$. Now note that both $\alpha_1^{\mathcal{N}}(a)$ and $\alpha_1^{\mathcal{N}}(c)$ are equal to $\emptyset$, and thus, in order to make the above diagram commute we must have both $[\widehat{\mathbf{R}}(\mathbf{y_0})](x)(\emptyset) = \{d\}$ and $[\widehat{\mathbf{R}}(\mathbf{y_0})](x)(\emptyset) = \emptyset$. This is clearly impossible for all functions $[\widehat{\mathbf{R}}(\mathbf{y_0})](x)$.

$$\begin{array}{ccc} \mathbf{y_0}(1) & \xrightarrow{\mathbf{y_0}(x)} & \mathbf{y_0}(1) \\ \downarrow{\scriptstyle \alpha_1^{\mathcal{N}}} & & \downarrow{\scriptstyle \alpha_1^{\mathcal{N}}} \\ [\widehat{\mathbf{R}}(\mathbf{y_0})](1) & \xrightarrow[[\widehat{\mathbf{R}}(\mathbf{y_0})](x)]{} & [\widehat{\mathbf{R}}(\mathbf{y_0})](1) \end{array}$$

Summarizing, there cannot exist such functor $\widehat{\mathbf{R}}$ because a and c perform the same transitions, while ax and cx perform different transitions.

The above example shows that usually we cannot define a coalgebra on presheaves that models the reaction relation of a reactive system. However, it is always possible to define these coalgebraic models for the saturated and the IPO transition systems.

Definition 20. The endofunctor $\mathbf{S} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ is defined as follows.
For all objects $\mathbf{X} \in |\mathbf{Set}^{\mathbf{C}}|$,
for all $i \in |\mathbf{C}|$, $[\mathbf{S}(\mathbf{X})](i) = \mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i,j] \times \mathbf{X}(j))$;
for all $i, j \in |\mathbf{C}|$ and $c \in \mathbf{C}[i,j]$, $[\mathbf{S}(\mathbf{X})](c) : [\mathbf{S}(\mathbf{X})](i) \to [\mathbf{S}(\mathbf{X})](j)$ maps

A into $\{(d, q) \mid (c; d, q) \in A\}$.

For all arrows $h \in ||\mathbf{Set}^{\mathbf{C}}||$, $\mathbf{S}(h) = \{\mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i,j] \times h_j) \mid i \in |\mathbf{C}|\}$.

Notice that the functor $\mathbf{S}$ simply extends the definition of $\mathbf{D}$ to the arrows of $\mathbf{C}$. Intuitively, an $\mathbf{S}$-coalgebra is a functor $\mathbf{X} : \mathbf{C} \to \mathbf{Set}$ (i.e., a $|\mathbf{C}|$-sorted set together with a function $\mathbf{X}(c)$ for each $c \in ||\mathbf{C}||$) and a natural transformation $\alpha : \mathbf{X} \Rightarrow \mathbf{S}(\mathbf{X})$. We can prove that the transition structure $\alpha_S^{\mathcal{R}}$ is always a natural transformation from $\mathbf{y_0}$ to $\mathbf{S}(\mathbf{y_0})$.

Proposition 5. *$\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle$ is an $\mathbf{S}$-coalgebra.*

Now consider the coalgebra $\langle \mathbf{y_0}, \alpha_I^{\mathcal{R}} \rangle$ corresponding to the IPO transition system. Usually it is not an $\mathbf{S}$ coalgebra, because $\alpha_I^{\mathcal{R}}$ is not a natural transformation from $\mathbf{y_0}$ to $\mathbf{S}(\mathbf{y_0})$.

Example 9. Here we show that $\alpha_I^{\mathcal{N}}$ ($\mathcal{N}$ is the reactive system of Example 3) is not a natural transformation from $\mathbf{y_0}$ to $\mathbf{S}(\mathbf{y_0})$.

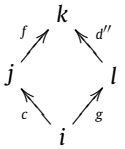
$$\begin{array}{ccc} \mathbf{OPN}_N[0,1] & \xrightarrow{-;y} & \mathbf{OPN}_N[0,1] \\ \downarrow{\scriptstyle \alpha_{I,1}^{\mathcal{R}}} & & \downarrow{\scriptstyle \alpha_{I,1}^{\mathcal{R}}} \\ \mathbf{P}_c(\mathbf{OPN}_N[1,1] \times \mathbf{OPN}_N[0,1]) & \xrightarrow[[\mathbf{S}(\mathbf{y_0})](y)]{} & \mathbf{P}_c(\mathbf{OPN}_N[1,1] \times \mathbf{OPN}_N[0,1]) \end{array}$$

Consider the IPOTS of cx and cxy of the open input petri net in Fig. 2(i). We have that $cx \xrightarrow{\varepsilon}_I d$ and $cxy \xrightarrow{\varepsilon}_I dy$. Thus $\alpha_I^{\mathcal{R}}(cx) = \{(\varepsilon, d)\}$ and $\alpha_I^{\mathcal{R}}(cxy) = \{(\varepsilon, dy)\}$. The latter is equal to $\alpha_I^{\mathcal{R}}(-; y)(cx)$, but $[\mathbf{S}(\mathbf{y_0})](y)(\alpha_I^{\mathcal{R}}(cx)) = \emptyset$. Indeed, by Definition 20, $[\mathbf{S}(\mathbf{y_0})](y)(\alpha_I^{\mathcal{R}}(cx)) = \{(u, v) \mid (y; u, v) \in \{(\varepsilon, d)\}\}$. Then, the above diagram does not commute.

In order to characterize the IPO transition system as a coalgebra on $\mathbf{Set}^{\mathbf{C}}$, we have to define a new endofunctor that behaves differently on the arrows of $\mathbf{C}$.

Definition 21. The endofunctor $\mathbf{I} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ is defined as follows.
For all objects $\mathbf{X} \in |\mathbf{Set}^{\mathbf{C}}|$,
for all $i \in |\mathbf{C}|$, $[\mathbf{I}(\mathbf{X})](i) = \mathbf{P}_c(\sum_{j \in |\mathbf{C}|} \mathbf{C}[i,j] \times \mathbf{X}(j))$;
for all $i, j \in |\mathbf{C}|$ and $c \in \mathbf{C}[i,j]$, $[\mathbf{I}(\mathbf{X})](c) : [\mathbf{I}(\mathbf{X})](i) \to [\mathbf{I}(\mathbf{X})](j)$ maps

A into $\{(f, \mathbf{X}(d'')(q)) \mid (g, q) \in A$ and the following is an IPO in $\mathbf{C}\}$.



For all arrows $h \in ||\mathbf{Set}^{\mathbf{C}}||$, $\mathbf{S}(h) = \{\mathbf{P}_c(\sum_{j\in|\mathbf{C}|} \mathbf{C}[i,j] \times h_j) \mid i \in |\mathbf{C}|\}$.

Proposition 6. *If $\mathcal{R}$ has redex-RPOs, then $\langle \mathbf{y_0}, \alpha_I^{\mathcal{R}} \rangle$ is an $\mathbf{I}$-coalgebra.*

Until now, we have characterized the saturated and the IPO transition systems as coalgebras on $\mathbf{Set}^{\mathbf{C}}$. In order to also characterize saturated and IPO bisimilarities as final semantics, we need the existence of final coalgebras.

Proposition 7. $\mathbf{Coalg_S}$ *and* $\mathbf{Coalg_I}$ *have final coalgebras.*

Since $\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle$ is an $\mathbf{S}$-coalgebra and since there exists a final coalgebra $1_{\mathbf{S}}$ in $\mathbf{Coalg_S}$, then the unique morphism $!^{\mathbf{S}}_{\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle} : \langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle \to 1_{\mathbf{S}}$ identifies all and only the saturated bisimilar states. Moreover, by Proposition 4, saturated bisimilarity is a congruence w.r.t. the arrows in $||\mathbf{C}||$. Analogously for IPO bisimilarity.

Corollary 2. *Let $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ be a reactive system.*
Then $\forall i \in |\mathbf{C}|$ and $\forall p, q \in \mathbf{C}[0, i]$:

1. *$p \sim^S q$ if and only if $!^{\mathbf{S}}_{\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle}(p) = !^{\mathbf{S}}_{\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle}(q)$,*
2. *if $p \sim^S q$ then $\forall j \in |\mathbf{C}|, \forall c \in \mathbf{C}[i, j]$ $p; c \sim^S q; c$.*

If $\mathcal{R}$ has redex-RPOs, then $\forall i \in |\mathbf{C}|$ and $\forall p, q \in \mathbf{C}[0, i]$:

3. *$p \sim^I q$ if and only if $!^{\mathbf{I}}_{\langle \mathbf{y_0}, \alpha_I^{\mathcal{R}} \rangle}(p) = !^{\mathbf{I}}_{\langle \mathbf{y_0}, \alpha_I^{\mathcal{R}} \rangle}(q)$,*
4. *if $p \sim^I q$ then $\forall j \in |\mathbf{C}|, \forall c \in \mathbf{C}[i, j]$ $p; c \sim^I q; c$.*

8. Conclusions

The main contribution of the paper is the appreciation of saturated semantics. These were already known in the literature [51,33], but were dismissed as not promising. In this paper (and, more extensively, in [6]) we have shown some interesting examples where saturated semantics coincide with the ordinary semantics, while IPO semantics are too fine-grained. The problem of the strictness of IPO semantics is also faced in [44–46] where, inspired by the theory of reactive systems, Rathke and Sobociński propose a new theory for labels derivations. As future work, we would like to investigate the relationship between this new theory and saturated semantics. Recent results shown in [9] prove that the two approaches coincide in the case of Mobile Ambients [16].

Since, by definition, saturated semantics consider all the contexts allowing a reaction (possibly infinitely many), we have proposed the semi-saturated game that allows to recover saturated semantics by considering only the minimal contexts. This technique has been introduced here for bisimilarity and (a kind of) trace equivalence, but it can be also applied to different equivalences. For example, in [9], it has been used for both *strong* and *weak barbed bisimilarity*. Moreover, in this paper we have integrated the semi-saturation into the IPO framework, but it could be applied also to G-reactive systems [50] and open reactive systems [31]. Besides the theory of reactive systems and the problem of labels derivations, semi-saturated bisimilarity can be employed for *symbolic semantics* [28]. In [13,6], we have shown that the symbolic semantics of some process calculi are instances of our semi-saturated bisimiliarity (properly extended to manage also labeled transitions). For example, open bisimilarity for π-calculus [48] is an instance of our saturated bisimilarity and its efficient characterization through symbolic semantics is an instance of our semi-saturated bisimilarity.

The coalgebraic models presented in the last part of the paper provide both a characterization of saturated and IPO bisimilarity as final semantics and an alternative proof of the fact that they are congruences. In this paper we have omitted the coalgebraic characterization of semi-saturated bisimilarity that was proposed in [12] by means of *normalized coalgebras*. These are a special kind of structured coalgebras [18] that, analogously to the semi-saturated technique, allow one to efficiently check saturated bisimilarity by considering only the IPO transitions. We are confident that investigating normalized coalgebras in the perspective proposed in this paper, we could define a general theory that relates coalgebras over presheaves (that elegantly model many nominal calculi) and normalized coalgebras (that allow one to efficiently check their abstract semantics).

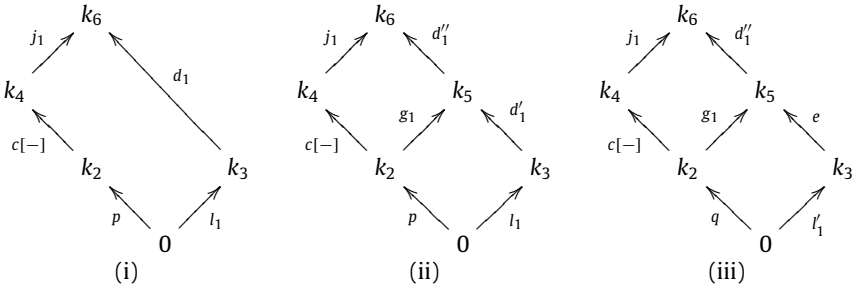
Appendix A. Proofs of Section 4.2

Proposition 2. *In a reactive system where all contexts are reactive and ϕ defines a composition-reflecting subcategory, $\simeq^{\phi}_{SAT}$ is a congruence.*

Proof. We show that $\{(c[p], c[q])$ s.t. $p \simeq^{\phi}_{SAT} q\} \subseteq \simeq^{\phi}_{SAT}$.
First of all, we have to prove that $\phi(c[p])$ if and only if $\phi(c[q])$. We prove that, if $\phi(c[p])$ then $\phi(c[q])$ (the other direction is analogous). Since ϕ defines a composition-reflecting subcategory, if $\phi(c[p])$ then $\phi(c)$ and $\phi(p)$. From the latter we derive that $\phi(q)$ (since $p \simeq^{\phi}_{SAT} q$) and from the former we derive $\phi(c[q])$.

Now, Suppose that $c[p] \overset{l}{\twoheadrightarrow}_S p' \wedge \phi(p')$, then $c[p] = p_1 \overset{l_1}{\rightarrow}_{SAT} p_2 \cdots p_n \overset{l_n}{\rightarrow}_{SAT} p_{n+1} = p'$ and $l = l_1; l_2; \ldots; l_n$. Then $p \overset{c[-];l_1}{\rightarrow}_{SAT} p_2$ and thus $p \overset{c[-];l}{\twoheadrightarrow}_S p'$. Since $p \simeq^{\phi}_{SAT} q$, then $q \overset{c[-];l}{\twoheadrightarrow}_S q'$ and $\phi(q')$. Because $c[-]$ is reactive, $c[q] \overset{l}{\twoheadrightarrow}_S q'$. □

Proposition 3. *In a reactive system with redex and context RPOs, where all contexts are reactive and ϕ defines a composition-reflecting subcategory, $\simeq^{\phi}_{l}$ is a congruence.*



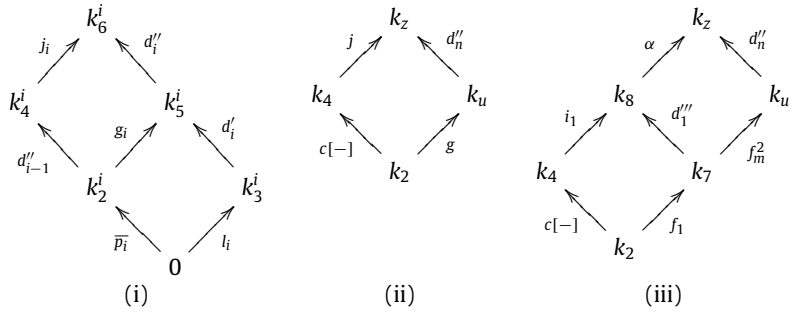
Proof. In order to prove this theorem we will use the composition and decomposition properties of IPOs, namely Proposition 2.1 and Proposition 2.2 of [33]. We have to prove that if $p \simeq^{\phi}_{l} q$ then $c[p] \simeq^{\phi}_{l} c[q]$.

Suppose that $c[p] \overset{j}{\twoheadrightarrow}_l p' \wedge \phi(p')$, then $c[p] = p_1 \overset{j_1}{\rightarrow}_l p_2 \ldots p_n \overset{j_n}{\rightarrow}_l p_{n+1} = p'$ and $j = j_1; j_2; \ldots; j_n$.

To make clear the presentation we first show what happens in the case that $n = 1$, and then we extend the reasoning for all n. Notice that this is not a proof by induction on n.

If $c[p] = p_1 \overset{j_1}{\rightarrow}_l p_2 = p'$ and $\phi(p')$, then there exists an IPO square like diagram (i) above, where $\langle l_1, r_1\rangle \in \mathfrak{R}$ and $p_2 = r_1; d_1$. Since, by hypothesis, the reactive systems has redex-RPOs, then we can construct an RPO as the one in diagram (ii) above. In this diagram, the lower square is an IPO, since RPOs are also IPOs (Proposition 1 of [33]). Since the outer square is an IPO and the lower square is an IPO, by IPO decomposition property, it follows that also the upper square is an IPO. Moreover, since ϕ defines a composition-reflecting subcategory, we have that $\phi(r_1), \phi(d'_1)$ and $\phi(d''_1)$. Now the lower square in diagram (ii) is an IPO, and then $p \overset{g_1}{\rightarrow}_l r_1; d'_1$ with $\phi(r_1; d'_1)$. Since $p \simeq^{\phi}_{l} q$, then $q \overset{g_1}{\rightarrow}_l q_2$ with $\phi(q_2)$. This means that the lower square in diagram (iii) is an IPO and $\langle l'_1, r'_1\rangle \in \mathfrak{R}$ and $q_2 = r'_1; e$. Notice that since $\phi(q_2)$, then $\phi(r'_1)$ and $\phi(e)$. Now, since the lower and the upper square of diagram (iii) are IPOs then, by IPO composition, also the outer square is an IPO. This means that $c[p] \overset{j_1}{\rightarrow}_l r'_1; e; d''_1$. Moreover, since $\phi(d''_1)$, then also $\phi(r'_1; e; d''_1)$.

We can repeat the procedure above for any n.



Suppose that $c[p] = p_1 \overset{j_1}{\rightarrow}_l p_2 \ldots p_n \overset{j_n}{\rightarrow}_l p_{n+1} = p'$ and $\phi(p')$. For all $i = 1 \ldots n$, by definition of $\overset{j_i}{\rightarrow}_l$, we have that the outer square in diagram (i) is an IPO, where $\overline{p_i}; d''_{i-1} = p_i$, $\langle l_i, r_i\rangle \in \mathfrak{R}$ and $r_i; d'_i; d''_i = p_{i+1}$. Moreover $\overline{p_1} = p$ and $d''_0 = c[-]$.

By decomposition property, the lower and the upper square in diagram (i) are IPOs and $\overline{p_{i+1}} = r_i; d'_i$. Therefore $p = \overline{p_1} \overset{g_1}{\rightarrow}_l \overline{p_2} \cdots \overline{p_n} \overset{g_n}{\rightarrow}_l \overline{p_{n+1}}$. Since $p_{n+1} = \overline{p_{n+1}}; d''_n$ and $\phi(p_{n+1})$, it holds that $\phi(d''_n)$ and $\phi(\overline{p_{n+1}})$. By composition property, the diagram (ii) is an IPO, because it is the composition of n squares as the upper square of diagram (i) that are all IPOs.

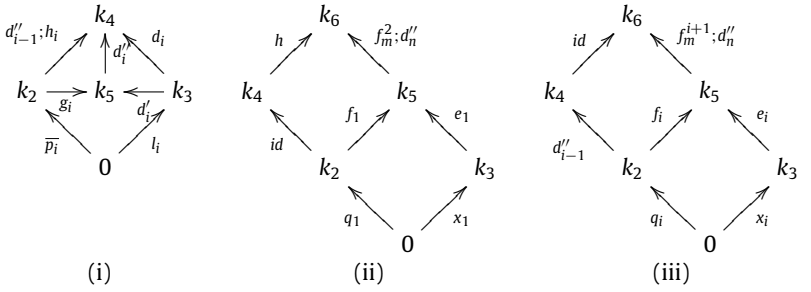
Let $g = g_1; g_2 \dots g_n$. Since $p \simeq_I^{\phi} q$ and $p \overset{g}{\twoheadrightarrow}_I \overline{p_{n+1}}$ with $\phi(\overline{p_{n+1}})$, then $q \overset{g}{\twoheadrightarrow}_I q'$ and $\phi(q')$. Unfortunately this does not mean that $q \overset{g_1}{\rightarrow}_I q_2 \cdots q_n \overset{g_n}{\rightarrow}_I q_{n+1}$, because g can be decomposed in many ways. So we have that exists $f_1, f_2, \dots, f_m$ such that $f_1; f_2; \dots; f_m = g$ and $q \overset{f_1}{\rightarrow}_I q_2 \dots q_m \overset{f_m}{\rightarrow}_I q_{m+1} = q'$.

Let $f_m^i = f_i; f_{i+1}; \dots; f_m$, then $g = f_1; f_m^2$. Since diagram (ii) commutes, and since by hypothesis the reactive system have context RPOs, we can construct the RPO between $c[-]$ and f_1, like diagram (iii) above, where $i_1; \alpha = j$. Since RPOs are IPO, the lower square of diagram (iii) is an IPO. Since, the outer square of diagram (iii) is an IPO (it is diagram (ii)), by decomposition property, it follows that also the upper square is an IPO. Since, $q \overset{f_1}{\rightarrow}_I q_2$ and since the lower square in diagram (iii) is an IPO, it follows that $c[p] \overset{i_1}{\rightarrow}_I q_2; d_1'''$.

Now, since the upper square in diagram (iii) is an IPO, we can iterate this procedure, obtaining that $c[q] \overset{i_1}{\rightarrow}_I q_2; d_1''' \dots q_m \overset{i_m}{\rightarrow}_I q_{m+1}; d_n''$. Since $j = i_1; i_2; \dots; i_m$, then $c[q] \overset{j}{\twoheadrightarrow} q_{m+1}; d_n''$ and $\phi(q_{m+1}; d_n'')$ because $\phi(q_{m+1})$ and $\phi(d_n'')$.

In order to complete the proof, we have to prove that $\phi(c[p])$ if and only if $\phi(c[q])$. This follows from the same arguments of the proof of Proposition 2. □

Theorem 4. *In a reactive system with redex IPOs, where all contexts are reactive, and such that ϕ defines a composition-reflecting subcategory, then $\simeq_{SS}^{\phi} = \simeq_{SAT}^{\phi}$.*



Proof. If $p \simeq_{SAT}^{\phi} q$ then, trivially, $p \simeq_{SS}^{\phi} q$.

For the other inclusion, let us consider the diagrams above. We suppose that $p \simeq_{SS}^{\phi} q$ and we prove that $p \overset{h}{\twoheadrightarrow}_S p' \wedge \phi(p')$ implies $q \overset{h}{\twoheadrightarrow}_S q' \wedge \phi(q')$. If $p \overset{h}{\twoheadrightarrow}_S p'$ then there exist $h_1, \dots, h_n$ such that $h_1; \dots; h_n = h$ and $p = p_1 \overset{h_1}{\rightarrow}_{SAT} p_2 \dots p_n \overset{h_n}{\rightarrow}_{SAT} p_{n+1} = p'$, and then $\exists \langle l_i, r_i \rangle \in \mathfrak{R}$, $d_i \in \mathbf{D}$ such that $p_i; h_i = l_i; d_i$ and $p_{i+1} = r_i; d_i$. Note that for all i diagram (i) commutes and the lower square is an IPO, where $\overline{p_1} = p$, $d_0'' = id$ and $p_i = \overline{p_i}; d_{i-1}''$. Then $p = \overline{p_1} \overset{g_1}{\rightarrow}_I \overline{p_2} \dots \overline{p_n} \overset{g_n}{\rightarrow}_I \overline{p_{n+1}}$. Since $p_{n+1} = \overline{p_{n+1}}; d_n''$ and $\phi(p_{n+1})$ then $\phi(\overline{p_{n+1}})$ and $\phi(d_n'')$. Let $g = g_1; \dots; g_n$, then $q \overset{g}{\twoheadrightarrow}_S q' \wedge \phi(q')$ and there exist $f_1, f_2, \dots, f_m$ such that $g = f_1; f_2; \dots; f_m$ and $q = q_1 \overset{f_1}{\rightarrow}_{SAT} q_2 \dots q_m \overset{f_m}{\rightarrow}_{SAT} q_{m+1} = q'$. By f_m^i we denote $f_i; f_{i+1}; \dots; f_m$. Note that $h = g; d_n''$,i.e., $h = f_1; f_m^2; d_n''$, and then $q \overset{h}{\rightarrow}_{SAT} q_2; f_m^2; d_n'' \overset{id}{\rightarrow}_{SAT} q_3; f_m^3; d_n'' \dots q_m; f_m^m; d_n'' \overset{id}{\rightarrow}_{SAT} q_{m+1}; d_n''$ (as illustrated in diagram (iii)). Indeed $\phi(q_{m+1}; d_n'')$ because $\phi(q_{m+1})$ and $\phi(d_n'')$.

In order to complete the proof, we have to prove that $\phi(c[p])$ if and only if $\phi(c[q])$. This follows from the same arguments of the proof of Proposition 2. □

Appendix B. Proofs of Section 7

Proposition 5. *$\langle \mathbf{y_0}, \alpha_S^{\mathcal{R}} \rangle$ is a $\mathbf{S}$-coalgebra.*

Proof. In order to prove the above proposition we must just show that $\alpha_S^{\mathcal{R}}$ is a natural transformation from $\mathbf{y_0}$ to $\mathbf{S}(\mathbf{y_0})$, i.e., we have to prove that $\forall i, j \in |\mathbf{C}|$ and $\forall c \in \mathbf{C}[i, j]$, the following diagram commutes.

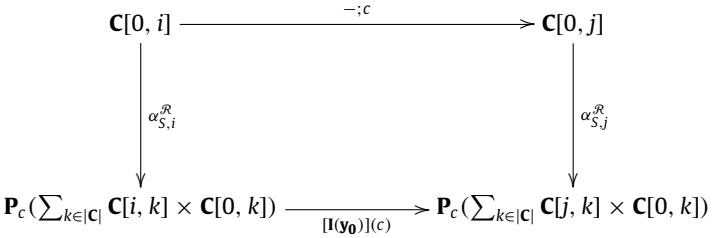
$$\begin{array}{ccc} \mathbf{C}[0,i] & \xrightarrow{\quad -;c \quad} & \mathbf{C}[0,j] \\ \Big\downarrow{\scriptstyle \alpha^{\mathcal{R}}_{S,i}} & & \Big\downarrow{\scriptstyle \alpha^{\mathcal{R}}_{S,j}} \\ \mathbf{P}_c\big(\sum_{k\in|\mathbf{C}|} \mathbf{C}[i,k] \times \mathbf{C}[0,k]\big) & \xrightarrow[[\mathbf{S}(\mathbf{y_0})](c)]{} & \mathbf{P}_c\big(\sum_{k\in|\mathbf{C}|} \mathbf{C}[j,k] \times \mathbf{C}[0,k]\big) \end{array}$$

Let $p \in \mathbf{C}[0,i]$. If $p; c \xrightarrow{d}_{SAT} q$ (i.e., $(d,q) \in \alpha^{\mathcal{R}}_{S,j}(p;c)$), then $p \xrightarrow{c;d}_{SAT} q$ (i.e., $(c;d,q) \in \alpha^{\mathcal{R}}_{S,i}(p)$). Now, by definition of $\mathbf{S}$, we have that $(d,q) \in [\mathbf{S}(\mathbf{y_0})](c)(\alpha^{\mathcal{R}}_{S,i}(p))$. This prove that $\alpha^{\mathcal{R}}_{S,j}(p;c) \subseteq [\mathbf{S}(\mathbf{y_0})](c)(\alpha^{\mathcal{R}}_{S,i}(p))$.

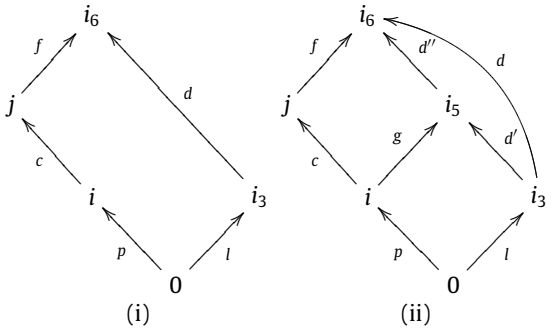
The other direction is analogous. □

Proposition 6. *If $\mathcal{R}$ has redex-RPOs, then $\langle \mathbf{y_0}, \alpha^{\mathcal{R}}_l \rangle$ is a $\mathbf{I}$-coalgebra.*

Proof. In order to prove the above proposition we must simply show that $\alpha^{\mathcal{R}}_l$ is a natural transformation from $\mathbf{y_0}$ to $\mathbf{I}(\mathbf{y_0})$, i.e., we have to prove that $\forall i,j \in |\mathbf{C}|$ and $\forall c \in \mathbf{C}[i,j]$, the following diagram commutes.



Let $p \in \mathbf{C}[0,i]$. If $p; c \xrightarrow{f}_l q$ (i.e., $(f,q) \in \alpha^{\mathcal{R}}_{l,j}(p;c)$), then there exists $d \in \mathbf{D}$, $\langle l, r \rangle \in \mathfrak{R}$ such that $q = r; d$ and diagram (i) below is an IPO.



Since diagram (i) is a redex square, and since by hypothesis $\mathcal{R}$ has redex-RPOs, we can construct an RPO as in diagram (ii). Since RPOs are IPOs (Proposition 1 of [33]) then the bottom square of diagram (ii) is an IPO. Moreover, by *IPOs decomposition* (Proposition 2.2 of [33]), we have that also the top square is an IPO. Thus $p \xrightarrow{g}_l r; d'$ (i.e., $(g, r; d') \in \alpha^{\mathcal{R}}_{l,i}(p)$). Since the top square is an IPO, by definition of $\mathbf{I}$, we have that $(f, r; d'; d'') \in [\mathbf{I}(\mathbf{y_0})](c)(\alpha^{\mathcal{R}}_{l,i}(p))$, i.e., that $(f,q) \in [\mathbf{I}(\mathbf{y_0})](c)(\alpha^{\mathcal{R}}_{l,i}(p))$.

For the other direction, we proceed as before but we use *IPO composition* (Proposition 2.1 of [33]) instead of decomposition. □

Proposition 7. $\mathbf{Coalg_S}$ *and* $\mathbf{Coalg_I}$ *have final coalgebras.*

Sketch of proof. This proposition was proved in [12] relying on the theory of *structured coalgebras* [18]. Differently from the well-known bialgebras [54], structured coalgebras are just coalgebras defined on a category of algebras $\mathbf{Alg}_\Gamma$ for some algebraic specification Γ.

We can look at the category $\mathbf{Set}^{\mathbf{C}}$ as the category $\mathbf{Alg}_{\Gamma(\mathbf{C})}$, where the specification $\Gamma(\mathbf{C})$ is defined as follow.

specification $\Gamma(\mathbf{C})$ =
  sorts
    i $\quad \forall i \in |\mathbf{C}|$
  operations
    $c : i \to j$ $\quad \forall c \in \mathbf{C}[i,j]$
  equations
    $id_i(x) = x$
    $e(d(x)) = c(x)$ $\quad \forall d; e = c.$

It is easy to see that every $\Gamma(\mathbf{C})$-algebra is a functor from $\mathbf{C}$ to $\mathbf{Set}$ and viceversa every functor from $\mathbf{C}$ to $\mathbf{Set}$ is a $\Gamma(\mathbf{C})$-algebra. The same can be said for morphisms. Thus, our functors $\mathbf{S} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ and $\mathbf{I} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ are also endofunctors on $\mathbf{Alg}_{\Gamma(\mathbf{C})}$. These are respectively, the functors $\mathbf{F}$ and $\mathbf{I}$ of [12], where we proved that final coalgebras exist for these functors.

Hereafter we recall the argument of that proof.

Consider the forgetful functor $\mathbf{V} : \mathbf{Alg}_{\Gamma(\mathbf{C})} \to \mathbf{Set}^{|\mathbf{C}|}$, that maps each $\mathbf{Alg}_{\Gamma(\mathbf{C})}$-algebra into its ($\mathbf{Alg}_{\Gamma(\mathbf{C})}$-sorted) carrier set. Now recall the functor $\mathbf{D} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ (Definition 19). It is easy to see that both $\mathbf{S}$ and $\mathbf{I}$ are *liftings* of $\mathbf{D}$ along the forgetful functor $\mathbf{V}$, i.e. $\mathbf{V}; \mathbf{D} = \mathbf{S}; \mathbf{V}$ and $\mathbf{V}; \mathbf{D} = \mathbf{I}; \mathbf{V}$.

Proposition 19 of [18] proves that, if an endofunctor $\mathbf{B}_\Gamma$ is a lifting (along a forgetful functor) of some functor $\mathbf{B}$ having final coalgebra, then also $\mathbf{B}_\Gamma$ has a final coalgebra.

From this proposition and the above observations, it follows that, if $\mathbf{D}$ has final coalgebra, then also $\mathbf{I}$ and $\mathbf{S}$ have final coalgebras.

To prove that $\mathbf{D}$ has final coalgebra we can use a well-known result from [1]: an endofunctor $\mathbf{B}$ on some category $\mathbf{C}$ has final coalgebra if $\mathbf{C}$ is locally finitely presentable and $\mathbf{B}$ is accessible.

Now, $\mathbf{D} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ has final coalgebra because $\mathbf{Set}^{|\mathbf{C}|}$ is locally finitely presentable (Example 1.12 of [1]) and it is accessible (by standard argument on accessibility of $\mathbf{P}_c$). □

References

[1] Jirí Adamek, Jirí Rosicky, Locally Presentable and Accessible Categories, vol. 189, Cambridge University Press, 1994.
[2] Roberto M. Amadio, Ilaria Castellani, Davide Sangiorgi, On bisimulations for the asynchronous pi-calculus, in: Proc. of CONCUR '96, in: LNCS, vol. 1119, Springer, 1996, pp. 147–162.
[3] Paolo Baldan, Andrea Bracciali, Roberto Bruni, Bisimulation by unification, in: Proc. of AMAST '02, in: LNCS, vol. 2422, Springer, 2002, pp. 254–270.
[4] Paolo Baldan, Andrea Corradini, Hartmut Ehrig, Reiko Heckel, Compositional semantics for open petri nets based on deterministic processes, Mathematical Structures in Computer Science 15 (1) (2005) 1–35.
[5] Gérard Berry, Gérard Boudol, The chemical abstract machine, Theoretical Computer Science 96 (1992) 217–248.
[6] Filippo Bonchi, Abstract semantics by observable contexts, Ph.D. Thesis, University of Pisa, 2008.
[7] Filippo Bonchi, Maria Grazia Buscemi, Vincenzo Ciancia, Fabio Gadducci, A category of explicit fusions, in: Concurrency, Graphs and Models, in: LNCS, vol. 5065, Springer, 2008, pp. 544–562.
[8] Filippo Bonchi, Fabio Gadducci, Barbara König, Process bisimulation via a graphical encoding, in: Proc. of ICGT '06, in: LNCS, vol. 4178, 2006, pp. 168–183.
[9] Filippo Bonchi, Fabio Gadducci, Giacoma V. Monreale, Labelled transitions for mobile ambients (as synthesized via a graphical encoding), in: Expressiveness in Concurrency, in: ENTCS, Elsevier, 2008 (forthcoming).
[10] Filippo Bonchi, Fabio Gadducci, Giacoma V. Monreale, Reactive systems, barbed semantics and the mobile ambients, in: Proc. of FoSSaCS '09, in: Lecture Notes in Computer Science, vol. 5504, Springer, 2009, pp. 272–287.
[11] Filippo Bonchi, Barbara König, Ugo Montanari, Saturated semantics for reactive systems, in: Logic in Computer Science, IEEE, 2006, pp. 69–80.
[12] Filippo Bonchi, Ugo Montanari, Coalgebraic models for reactive systems, in: Proc. of CONCUR '07, in: LNCS, vol. 4701, Springer, 2007, pp. 364–380.
[13] Filippo Bonchi, Ugo Montanari, Symbolic semantics revisited, in: Proc. of FoSSaCS '08, in: LNCS, vol. 4962, Springer, 2008, pp. 395–412.
[14] Roberto Bruni, Fabio Gadducci, Ugo Montanari, Pawel Sobociński, Deriving weak bisimulation congruences from reduction systems, in: Proc. of CONCUR '05, in: LNCS, vol. 3653, Springer, 2005, pp. 293–307.
[15] Roberto Bruni, Ugo Montanari, Francesca Rossi, An interactive semantics of logic programming, Theory and Practice of Logic Programming 1 (6) (2001) 647–690.
[16] Luca Cardelli, Andrew D. Gordon, Mobile ambients, Theoretical Computer Science 240 (1) (2000) 177–213.
[17] Gian Luca Cattani, Peter Sewell, Models for name-passing processes: Interleaving and causal, in: Logic in Computer Science, IEEE.
[18] Andrea Corradini, Martin Große-Rhode, Reiko Heckel, A coalgebraic presentation of structured transition systems, Theoretical Computer Science 260 (2001) 27–55.
[19] Andrea Corradini, Reiko Heckel, Ugo Montanari, From sos specifications to structured coalgebras: How to make bisimulation a congruence, ENTCS 19 (1999).
[20] Pietro di Gianantonio, Furio Honsel, Marina Lenisa, Rpo, second-order contexts, and λ-calculus, in: Proc. of FoSSaCS '08, in: LNCS, vol. 4962, Springer, 2008, pp. 334–349.
[21] Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski, Ugo Montanari, Grzegorz Rozenberg (Eds.), Handbook of Graph Grammars and Computing by Graph Transformation, vols. 1–3, World Scientific, 1997–1999.
[22] Moreno Falaschi, Giorgio Levi, Catuscia Palamidessi, Maurizio Martelli, Declarative modeling of the operational behavior of logic languages, Theoretical Computer Science 69 (3) (1989) 289–318.
[23] Marcelo P. Fiore, Eugenio Moggi, Davide Sangiorgi, A fully abstract model for the pi-calculus, Information and Computation 179 (1) (2002) 76–117.
[24] Marcelo P. Fiore, Daniele Turi, Semantics of name and value passing, in: Logic in Computer Science, IEEE, 2001, pp. 93–104.
[25] Neil Ghani, Kidane Yemane, Björn Victor, Relationally staged computations in calculi of mobile processes, ENTCS 106 (2004) 105–120.
[26] J. Goguen, What is unification? A categorical view of substitution, equation and solution, in: Resolution of Equations in Algebraic Structures, 1989, pp. 217–261.
[27] Davide Grohmann, Marino Miculan, Reactive systems over directed bigraphs, in: Proc. of CONCUR '07, in: LNCS, vol. 4703, Springer, 2007, pp. 380–394.
[28] Matthew Hennessy, H. Lin, Symbolic bisimulations, Theoretical Computer Science 138 (2) (1995) 353–389.
[29] Ole H. Jensen, Robin Milner, Bigraphs and transitions, in: POPL, IEEE, 2003, pp. 38–49.
[30] Ekkart Kindler, A compositional partial order semantics for petri net components, in: Proc. of ATPN '97, in: LNCS, vol. 1248, 1997, pp. 235–252.
[31] Bartek Klin, Vladimiro Sassone, Pawel Sobociński, Labels from reductions: Towards a general theory, in: Proc. of CALCO '05, in: LNCS, vol. 3629, Springer, 2005, pp. 30–50.
[32] F. William Lawvere, Some algebraic problems in the context of functorial semantics of algebraic theories, in: Proc. of the Midwest Category Seminar II, in: Lecture Notes in Mathematics, vol. 61, 1968, pp. 41–61.
[33] James J. Leifer, Robin Milner, Deriving bisimulation congruences for reactive systems, in: Proc. of CONCUR '00, in: LNCS, vol. 1877, Springer, 2000, pp. 243–258.
[34] James J. Leifer, Robin Milner, Transition systems, link graphs and petri nets, Mathematical Structures in Computer Science 16 (6) (2006) 989–1047.
[35] Massimo Merro, Francesco Zappa Nardelli, Behavioral theory for mobile ambients, Journal of the ACM 52 (6) (2005) 961–1023.
[36] Marino Miculan, Kidane Yemane, A unifying model of variables and names, in: Proc. of FoSSaCS '05, in: LNCS, vol. 3441, Springer, 2005, pp. 170–186.
[37] Robin Milner, Bigraphical reactive systems, in: Proc. of CONCUR '01, in: LNCS, vol. 2154, Springer, 2001, pp. 16–35.
[38] Robin Milner, Bigraphs for petri nets, in: Lectures on Concurrency and Petri Nets, in: LNCS, vol. 3098, Springer, 2004, pp. 686–701.
[39] Robin Milner, Pure bigraphs: Structure and dynamics, Information and Computation 204 (2006) 60–122.
[40] Robin Milner, Local bigraphs and confluence: Two conjectures, ENTCS 175 (3) (2007) 65–73.
[41] Robin Milner, Joachim Parrow, David Walker, A calculus of mobile processes, (i) and (ii), Information and Computation 100 (1) (1992) 1–40, 41–77.
[42] Robin Milner, Davide Sangiorgi, Barbed bisimulation, in: Proc. of ICALP '92, in: LNCS, vol. 623, Springer, 1992, pp. 685–695.
[43] Ugo Montanari, Vladimiro Sassone, Dynamic congruence vs. progressing bisimulation for ccs, Fundamenta Informaticae 16 (1) (1992) 171–199.
[44] Julian Rathke, Pawel Sobocinski, Deconstructing behavioural theories of mobility, in: Theoretical Computer Science, in: IFIP, vol. 273, Springer, 2008.
[45] Julian Rathke, Pawel Sobociński, Deriving structural labelled transitions for mobile ambients, in: Proc. of CONCUR '08, in: LNCS, vol. 5201, Springer, 2008.
[46] Julian Rathke, Pawel Sobocinski, Making the unobservable, unobservable (2008) (forthcoming).
[47] Jan J.M.M. Rutten, Universal coalgebra: A theory of systems, Theoretical Computer Science 249 (1) (2000) 3–80.
[48] Davide Sangiorgi, A theory of bisimulation for the pi-calculus, Acta Informatica 33 (1) (1996) 69–97.

[49] Vladimiro Sassone, Pawel Sobociński, A congruence for Petri nets, in: Petri Nets and Graph Transformation, in: ENTCS, vol. 127, Elsevier, 2005, pp. 107–120.
[50] Vladimiro Sassone, Pawel Sobociński, Locating reaction with 2-categories, Theoretical Computer Science 333 (1–2) (2005) 297–327.
[51] Peter Sewell, From rewrite to bisimulation congruences, in: Proc. of CONCUR '98, in: LNCS, vol. 1466, Springer, 1998, pp. 269–284.
[52] Pawel Sobociński, Deriving process congruences from reaction rules, Ph.D. Thesis, University of Aarhus, 2004.
[53] Ian Stark, A fully abstract domain model for the π-calculus, in: Logic in Computer Science, IEEE, 1996, pp. 36–42.
[54] Daniele Turi, Gordon D. Plotkin, Towards a mathematical operational semantics, in: Logic in Computer Science, IEEE, 1997, pp. 280–291.
[55] Lucian Wischik, Philippa Gardner, Explicit fusions, Theoretical Computer Science 340 (3) (2005) 606–630.