Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Adequacy Issues in Reactive Systems: Barbed Semantics for Mobile Ambients

Giacoma Monreale

Supervisor
Fabio Gadducci

May 24, 2011

Abstract

Reactive systems represent a meta-framework aimed at deriving behavioral congruences for those specification formalisms whose operational semantics is provided by rewriting rules.

The aim of this thesis is to address one of the main issues of the framework, concerning the adequacy of the standard observational semantics (the IPO and the saturated one) in modelling the concrete semantics of actual formalisms. The problem is that IPO-bisimilarity (obtained considering only minimal labels) is often too discriminating, while the saturated one (via all labels) may be too coarse, and intermediate proposals should then be put forward.

We then introduce a more expressive semantics for reactive systems which, thanks to its flexibility, allows for recasting a wide variety of observational, bisimulation-based equivalences. In particular, we propose suitable notions of barbed and weak barbed semantics for reactive systems, and an efficient characterization of them through the IPO-transition systems.

We also propose a novel, more general behavioural equivalence: L-bisimilarity, which is able to recast both its IPO and saturated counterparts, as well as the barbed one. The equivalence is parametric with respect to a set L of reactive systems labels, and it is shown that under mild conditions on L it is a congruence.

In order to provide a suitable test-bed, we instantiate our proposal over the asynchronous CCS and, most importantly, over the mobile ambients calculus, whose semantics is still in a flux.

To Marco and Chiara: my dreams come true...

Acknowledgments

First of all, I would like to thank my supervisor Fabio Gadducci, who constantly supported, guided and encouraged me during these years of research. Working with him was a pleasure and an honour.

I am grateful to Filippo Bonchi for our collaboration: it allowed me to learn so much, building at the same time a friendship.

I wish to thank Ugo Montanari, who conveyed to me the passion for theoretical computer science in all the occasions we discussed together. I also thank him and Giorgio Ghelli for their patience in overseeing my work as members of the supervisory committee.

I thank the reviewers for their useful and constructive suggestions, which led to several improvements of the thesis.

I would also like to express my gratitude to all the people, among them Paolo Baldan, with whom I had the pleasure to work together or/and I had the opportunity to exchange ideas: it always allowed me to have fresh insights and novel points of view.

A special thought goes to my colleague and close friend Mauriana Pesaresi. Her smile shall always be in my mind and in my heart.

Many thanks to my family, and in particular to my sisters Anna and Marcella, for always believing in me. Special thanks go to Anna, who gave me a very beautiful niece, Sofia. I would also like to thank her for sharing with me the joys and the difficulties of these important years of study and life. I hope to have the opportunity to work again with her.

I owe a lot to my husband Marco and my wonderful daughter Chiara, who lovingly bear with me during the writing of this thesis and made me laugh during my breaks. I think I can not write all I would like to tell them, because these acknowledgments would inevitably become a love letter.

Contents

List of Figures

Chapter 1

Introduction

The ever increasing diffusion of concurrent and distributed systems stimulated the development of novel formalisms for their specification. Roughly, on the one side we have more classical, syntax-based frameworks such as those related to process calculi; on the other side we witness the renewed interest towards visual models based on graph rewriting. Nowadays, these formalisms usually provide an abstract presentation of the behaviour of a system by resorting to some kind of operational description, eventually exploiting an observational equivalence.

Reduction semantics. At its simplest, the dynamics of a computational model is defined by means of a reduction semantics [1]: a set representing the possible states of the system, plus an unlabelled relation among these states, called reduction relation. The set of possible states is often provided by means of an equational specification, denoted as "structural congruence" in the process calculi literature, stating which presentations intuitively specify the same system, up-to a syntactical rearrangement of its components. The reduction relation, usually denoted by $\rightarrow$, describes the evolution of systems over time: $P \rightarrow Q$ means that the state P reduces to Q, that is, P can execute a computational step and it is transformed into Q. The reduction relation is closed under structural congruence and it is inductively generated by a set of axioms and a set of structural rules, which close the relation under some contexts. A reduction rule is a pair $\langle l, r \rangle$, where l represents the left hand side of the rule and r is the right hand side. So, a state P reduces into a state Q, $P \rightarrow Q$, if the left hand side l of a reduction rule occurs in it, that is, $P = C[l]$. In this case, the left hand side is replaced by the right hand side r and therefore $Q = C[r]$.

For example, the reduction rule modelling the (asynchronous) CCS-like communication over a channel a is $a.P \mid \bar{a} \rightarrow P$. Intuitively, the rule says that a process sending a message on a channel a and a process receiving on the same channel can react by consuming the two actions and continuing as P. So, the operational semantics of the process $a.b.\mathbf{0} \mid \bar{a} \mid \bar{c}$ is obtained by instantiating the above rule to $b.\mathbf{0}$ and contextualizing it in the unary context $- \mid \bar{c}$, hence obtaining the reduction $a.b.\mathbf{0} \mid \bar{a} \mid \bar{c} \rightarrow b.\mathbf{0} \mid \bar{c}$.

Despite the advantage of conveying the semantics with relatively few compact rewriting rules, the main drawback of reduction semantics is that it may be quite hard to devise meaningful behavioral equivalences (i.e., state equivalences based on the possible behaviour of systems), and more so if they are required to be congruences (that is, closed with respect to all the contexts of the specification). Being a congruence is a desirable property, since it allows one to replace a subsystem with an equivalent one without changing the behaviour of the overall system, thus stimulating the need of defining similar equivalences for reduction-based formalisms.

Barbed semantics. Various attempts of defining compositional behavioral equivalences starting from a reduction semantics have been made. An intuitive proposal already appearing in the literature on functional languages [4], and further expoited in the field of process calculi [54], is based on so-called *barbed equivalences*, uniformly describing an equality between systems specified by using calculi equipped with a reduction relation and a notion of predicate, called *barb*, which usually detects the possibility of performing some action. In particular, in [54] the authors take into account CCS [48], by showing that for that calculus the congruence induced by so-called *barbed bisimulation* coincides with standard, strong bisimulation.

[1]The reduction semantics is also called reduction semantics, so from now on we will use them indifferently.

However, the framework proved successful also for other calculi as well (and we just mention here the case of mobile ambients [16], as reported in [47]), each time adopting an ad-hoc notion of barb, specific for the calculus at hand. However, these equivalences often require the quantification over all contexts, so proofs of system equivalence can be very complex and hard to provide.

Labelled transition systems. An alternative approach equips the computational formalisms with an observational semantics by adding a *label* to each reduction. A labelled transition system (LTS) consists of a set of states and a labelled transition relation, i.e., an indexed relation among states, describing how the system can interact with the environment. So, for instance, a transition with label α from a state P to the state Q, in symbols $P \xrightarrow{\alpha} Q$, means that P can evolve to Q by interacting with the external environment, and the label α describes the interaction. Labels are exploited to define more abstract semantics by identifying systems with the same observational behaviour, in order to abstract aspects of system behaviour which should be ignored. Often, these observational equivalences (let them be alternative variants such as trace, testing, bisimulation $\ldots$) are compositional, that is, they are congruences with respect to all the possible contexts of the specification.

These labelled semantics are usually less intuitive than the reduction ones, and it might be difficult to identify the intuitively correct LTS specifying a given formalism. A case at hand is the calculus of *mobile ambients* [16], for which only recently suitable labelled semantics were proposed [47, 60], while for example for π-calculus [49], there exist at least two main LTSs, the early and the late version, giving rise to different behavioural equivalences.

Deriving bisimulation congruences from reductions. A series of papers recently addressed the need to derive LTSs starting from a reduction semantics, in order to derive observational equivalences (and more specifically, bisimulation equivalences [48, 56]) that are also congruences. The most successful technique adopted so far is represented by the theory of *reactive systems* [45]. It is based on so-called *relative pushouts* (RPOs), capturing in an abstract setting the intuitive notion of "minimal" environment into which a system specification has to be inserted, in order to allow a reduction to occur. The idea is very simple: whenever a system specified by a term $C[P]$, i.e., by a subterm P inserted into a (unary) "minimal" context $C[-]$, may evolve to a state Q, the associated LTS has a transition $P \xrightarrow{C[-]} Q$, i.e., the state P evolves into Q with a label $C[-]$. The resulting behavioural equivalence, called *IPO-bisimilarity*, is a congruence if "enough" RPOs exist.

Should all the possible contexts allowing a reduction be admitted, the resulting equivalence, denoted as *saturated bisimilarity*, would also result in a congruence. However, it is usually untractable, since it has to tackle a potentially infinite set of contexts. The problem has been addressed in [13] by introducing an "efficient" characterization (so-called semi-saturation) of these semantics, where one avoids considering all possible contexts by using in a cunning way RPOs, at the price of modifying the standard, symmetric presentation of the (either strong or weak) bisimulation equivalences.

In any case, providing the proof that e.g. a process calculus satisfies the requirements needed for applying the RPOs technique is often quite a daunting task, due to the intricacies of the structural congruence. A way out of the impasse is to look for graphical encodings of processes, such that process congruence is turned into graph isomorphism. Graph formalisms are more amenable to the RPOs trappings, and once the processes of a calculus have been encoded as graphs, a suitable LTS can thus be distilled. Indeed, the main source of examples concerning RPOs have been *bigraphs* [51], a graphical formalism introduced for specifying concurrent and distributed systems.

It is noteworthy that, should the reduction relation over graphs be defined using the double pushout (DPO) approach [2], these graphs are amenable to the *borrowed contexts* (BCs) technique [28], which offers a constructive solution for calculating the minimal contexts enabling a graph transformation rule. Indeed, graphs form an *adhesive* category [44], and for these formalisms borrowed context and RPO may be proved to be coincident notions [63].

1.1 Thesis Contribution

A less explored, yet a key issue in the theory of reactive systems concerns the adequacy of the observational semantics associated to the distilled LTS. As discussed in [6], IPO-bisimilarity is often too strict (it identifies less systems than expected), while the saturated one may be too coarse. As a paradigmatic case, the standard strong bisimilarity for CCS [48] coincides with the IPO-bisimilarity, and it is strictly included in the saturated one [7]; while for the asynchronous version of the calculus [31] IPO-bisimilarity does not capture the standard semantics (see below).

Barbed semantics for reactive systems. From a theoretical point of view, possibly the main technical contribution of this thesis is the introduction of suitable notions of (weak) barbed semantics for reactive systems, and their efficient characterization via transition systems labelled with minimal contexts, by exploiting the semi-saturated game. In order to properly establish the adequacy of the framework, we check it against suitable case studies. To this end, we instantiate our proposal over the asynchronous CCS, and most importantly, over the calculus of mobile ambients, whose observational semantics is still in a flux. In particular, for the asynchronous CCS, we show that strong barbed semantics is able to capture the standard asynchronous bisimilarity for the calculus, while for mobile ambients, we prove that the strong and the weak reduction barbed congruence, proposed respectively in [60], and in [47], coincide with the strong and weak barbed semantics for the calculus.

A more general behavioural equivalence for reactive systems. After the introduction of the more expressive barbed semantics for reactive systems, we take a step forward, by proposing a novel behavioural equivalence: L-bisimilarity. The equivalence is so called because it is parametric with respect to a set L of minimal labels and we show that under mild conditions on L it is a congruence. The equivalence is intermediate between its IPO and saturated counterparts: indeed, it is able to recover both of them, by simply varying the set of labels L. Furthermore, L-bisimilarity can also recast the notion of barbed semantics for reactive systems discussed above. With respect to the barbed case, L-bisimilarity admits a streamlined definition, where state predicates play no role. It is thus of simpler verification, and its introduction may have far reaching consequences over the usability of the reactive systems formalism. In order to provide a suitable test-bed, we instantiate our proposal again by addressing the semantics of the asynchronous CCS and of the calculus of mobile ambients.

Graphical encodings. For mobile ambients as well as asynchronous CCS, in order to identify the set L of minimal labels we exploit two minimal LTSs distilled by means of graphical encodings. In order to perform such a synthesis, processes are mapped into standard graphs such that process congruence is turned into graph isomorphism, while the reduction relation over processes is captured by a set of graph rewriting rules. In particular, while the graphical encoding for the asynchronous CCS is an adaptation of the one for the synchronous version proposed in [7], we present a novel encoding for the mobile ambients calculus, discussing its differences and advantages with respect to alternative proposals in literature, and providing an in-depth study. We also discuss the concurrency features of the proposed graph transformation system and we show how the information about dependencies among (causally related) rewriting steps offered by the graph-based semantics of mobile ambients may be used to identify interferences between process reductions, formalising the taxonomy proposed in [46].

Minimal LTSs via graphical encodings. Graphical encodings for mobile ambients and asynchronous CCS, are used to distill LTSs on (processes encoded as) graphs, by applying the borrowed context mechanism, hence, an instance of the RPO technique. For each calculus, we then use the synthesized LTS in order to infer a set of rules that is directly defined on the processes. As far as the mobile ambients calculus is concerned, we also propose an alternative, yet equivalent presentation of that LTS, by means of a set of structural rules, and we prove that it is the same as the one previously proposed in [60].

1.2 Outline of the Thesis

Chapter 2. Background on reactive systems: In this chapter we aim at giving a general introduction to the theory of reactive systems [45] and its extension to 2-dimensional categories [61]. We also introduce the borrowed context technique [28], addressing the problem of deriving labelled transitions systems from unlabelled reduction rules in the context of the double-pushout (DPO) approach to graph rewriting. Finally, a sketch of the connection between the two approaches is reported, as devised in [63].

Chapter 3. Graphical encodings for mobile ambients and asynchronous CCS: In this chapter we present the graphical encodings for the two calculi on which we test the main results presented in the thesis, namely, mobile ambients [16] and asynchronous CCS. In particular, here we briefly introduce the two calculi and for both of them we define an encoding mapping processes into graphs, showing its soundness and completeness with respect to the reduction semantics of the calculus. Such encodings will be then used in the next chapter for the synthesis of minimal labelled transition systems over graphs.

Much of the content of this chapter appeared first in the conference paper [34] and then in its journal version [35]: only Section 3.9 can be found in the conference paper [12].

Chapter 4. RPO semantics for mobile ambients and asynchronous CCS: In this chapter we apply the borrowed contexts technique to the two graphical encodings presented in the previous chapter. In particular, for both mobile ambients and asynchronous CCS, we present a suitable LTS directly defined over the structure of processes, obtained analyzing the synthesized LTS on graphs. These LTSs will be essential in showing the adequacy of the results presented in the two following chapters.

The development about the synthesis of the labelled transition systems for mobile ambients first appeared in the workshop paper [9] and in its submitted journal version [8]. The part concerning the asynchronous CCS can be found in [10] and [12].

Chapter 5. Barbed semantics for reactive systems: In this chapter we provide a framework for recasting (weak) barbed equivalence in the reactive systems formalism. We prove that our proposal captures the behavioural semantics for mobile ambients proposed in [60] and [47], as well as the standard semantics for asynchronous CCS. To this end we exploit the minimal contexts semantics for these calculi presented in Chapter 4.

Most of the results presented in this chapter were published in the conference paper [11].

Chapter 6. On barbs and labels in reactive systems: In this chapter we present a new, more general semantics for reactive systems, namely L-semantics, which is able to capture both its minimal and saturated counterparts, as well as, under suitable conditions, the more expressive barbed semantics. We test the proposed framework on the case studies, by showing that our proposal is able to capture the standard semantics for the mobile ambients and asynchronous CCS.

Results of this chapter appeared in the workshop paper [10].

Chapter 7. Conclusions: In this chapter we summarize the main results of the thesis and sketch possible future lines of research.

At the end of the thesis there are three technical Appendices, namely A, B and C, where we show the proofs of the results presented in Chapters 3, 4 and 5, respectively.

Chapter 2

Background on reactive systems

This chapter aims at giving a general introduction to the theory of reactive systems (Section 2.1), and its 2-categorical extension (Section 2.2). The borrowed context technique is also presented, and a sketch of its relationship with the general framework included (Section 2.3). In the presentation of the chapter we assume some elementary knowledge of the basic notions of category theory (pushouts, pullbacks, . . .).

2.1 The Theory of Reactive Systems

This section summarizes the main results concerning (the theory of) reactive systems introduced by Leifer and Milner [45]. The aim of the formalism is deriving labelled transition systems for those specification formalisms whose operational semantics is provided by reduction rules, such that the associated bisimulation equivalence is a congruence.

The framework is centered on the concepts of *context*, *term* and *reduction rules*.

Roughly, a context is a term with a hole. Given a context $C[-]$ and a term t, we would like to plug t into $C[-]$ and obtain the term $C[t]$. Moreover, given two contexts $C[-]$ and $D[-]$, we would like to compose them by substituting for example the first context within the second one so obtaining the new context $D[C[-]]$. This substitution operation between contexts should be also associative. This means that the context we obtain by inserting $C[-]$ inside $D[-]$ and then the result $D[C[-]]$ inside $E[-]$ should coincide with the term obtained by first inserting $D[-]$ inside $E[-]$ and then plugging the context $C[-]$ in the resulting context $E[D[-]]$. Moreover, we would like to have an identity context $-$, such that the context obtained by plugging a context $C[-]$ into $-$ is exactly $C[-]$ itself. It is therefore quite natural to model contexts as arrows of a category where composition of arrows is composition of contexts, and objects describe the types of such contexts.

In order to model a term as a context with no hole, we need a *distinguished* object 0 of the category, which is a special object denoting the lack of holes, such that arrows having 0 as domain represent terms.

Now, the last concept we need to introduce before giving the formal definition of reactive systems is the one of reduction rules. They are pairs of (ground) terms (arrows with domain 0) $\langle l, r\rangle$, where l is the left-hand side of the rule and r is the right-hand side. The reduction relation is hence obtained by closing them under certain contexts, called *reactive contexts*. Indeed, in general, there might be contexts inside which reductions cannot occur. For example, if we consider the Milner's CCS, we have that $b \mid \bar{b} \rightarrow \mathbf{0}$ yet $a.(b \mid \bar{b})$ has no reduction.

Now, we are ready to introduce the definition of reactive system. Given a category $\mathbf{C}$, we denote the class of morphisms with source m and target n by $\mathbf{C}(m, n)$.

Definition 2.1 (Reactive System)**.** *A* reactive system $\mathbb{C}$ *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* reduction rules.

By composition-reflecting we mean that $d; d' \in \mathbf{D}$ implies $d, d' \in \mathbf{D}$. Note that the left-hand and right-hand sides of reduction rules have the same codomain. This allows to obtain the definition of reduction relation which is generated from the reduction rules by closing them under all reactive contexts.

Definition 2.2 (Reduction Relation)**.** *Given a reactive system $\langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$, the reduction relation $\rightarrow$ is defined as follows: $P \rightarrow Q$ iff $P = l; d$ and $Q = r; d$ for some reduction rule $\langle l, r \rangle \in \mathfrak{R}$ and reactive context $d \in \mathbf{D}$ (Figure 2.1).*

$$\begin{tikzcd}
 & I_2 & \\
 & I_1 \arrow[u, "d"'] & \\
 & 0 \arrow[u, bend left, "l"] \arrow[u, bend right, "r"'] \arrow[uu, bend left=70, "P"] \arrow[uu, bend right=70, "Q"'] &
\end{tikzcd}$$

Figure 2.1: Reduction relation

Example 2.1. Let us consider the Simple Process Calculus (SPC), a trivial subset of the Milner's CCS proposed in [64]. The first row of Figure 2.2 shows the syntax of the calculus. We assume a set $\mathcal{N}$ of names ranging over by $a, b, c, \ldots$, and we let $P, Q, R, \ldots$ range over the set of processes.

The processes are considered up to the structural congruence ($\equiv$) induced by the only axioms in the second row of Figure 2.2. In order to keep the example as simple as possible, no structural rule guaranteeing that the **0** process is the identity for parallel composition is added.

The transition relation $\rightarrow$ is defined by the rules in the bottom of the same figure. The first axiom on the left models the synchronization over a channel a. The middle and the right rules instead model the closure of the relation with respect to the parallel composition and the structural congruence.

In the following we define the corresponding reactive system $\mathbb{C}_{SPC} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$.

The category $\mathbf{C}$. It has only 0 and 1 as objects, and terms over the signature $\Sigma = \mathbf{0} : 0, a : 0, \bar{a} : 0, | : 2$ (corresponding to the SPC grammar) as arrows. Terms are considered quotiented by the associativity and commutativity equations (second row of Figure 2.2) . In particular, the homset $\mathbf{C}(0, 0)$ contains only the identity arrow. There are no arrows from 1 to 0. Arrows of $\mathbf{C}(0, 1)$ represent ground terms, while arrows of $\mathbf{C}(1, 1)$ represent contexts, that is, terms with just one hole.

Note that the composition of arrows models the substitution of a context in the unique hole within another context. In particular, two types of substitution are allowed: the composition between a closed term $P : 0 \rightarrow 1$ and a context $C[-] : 1 \rightarrow 1$ resulting in the closed term $C[P] : 0 \rightarrow 1$, and the composition between a context $C[-] : 1 \rightarrow 1$ and a context $D[-] : 1 \rightarrow 1$ resulting in the context $D[C[-]] : 1 \rightarrow 1$.

The distinguished object. The object 0 denoting the lack of hole.

$$P ::= \mathbf{0}, a, \bar{a}, P \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad P \mid Q \equiv Q \mid P$$

$$a \mid \bar{a} \rightarrow \mathbf{0} \qquad \frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R} \qquad \frac{P' \equiv P, P \rightarrow Q, Q \equiv Q'}{P' \rightarrow Q'}$$

Figure 2.2: Syntax, structural congruence and reduction relation of SPC.

Reactive contexts. All contexts are reactive.

Reduction rules. The set $\{\langle a \mid \bar{a}, \mathbf{0}\rangle \mid a \in \mathcal{N}\}$.

The behaviour of a reactive system is expressed as an unlabelled transition system. As previously said, this kind of semantics is very natural and intuitive, but unfortunately it is not compositional. For example, consider the two SPC processes a and b, neither of them can perform a transition, so since they have the same operational behavior they are considered equivalent. If we insert them into the context $- \mid \bar{a}$, the former has a transition because it can execute the synchronization over the channel a, instead the latter has no transition. So, the equivalence is not preserved and hence it is not a congruence.

Labelled transition systems represent an alternative method used to give the operational semantics of formalisms, often inducing compositional behavioural equivalences.

Figure 2.3 shows the LTS for the fragment of the CCS introduced in Example 2.1. We adopt the same presentation used in [64]: the rules of the LTS are indeed not in the standard SOS style [57] and there is also an explicit rule closing the transition relation under the structural congruence. The first two axioms model the execution respectively of an input and an output over a channel. The middle axiom instead models the internal computation, while the last two rules model the closure of the labelled transition relation with respect to the parallel composition and the structural congruence, respectively.

If we consider again the two SPC processes above, that is, a and b, we can easily note that the LTS semantics above allows us to immediately distinguish them. The former indeed has the labelled transition $a \xrightarrow{a} \mathbf{0}$, while the latter has no transition labelled with a, but just one labelled with b, $b \xrightarrow{b} \mathbf{0}$.

Intuitively, the transition $a \xrightarrow{a} \mathbf{0}$ tell us that the process a can interact by performing an input over the channel a. In this case the label of the transition reflects an agent's capability to perform a certain action. If we shift our attention from the agent's capability to the context allowing the agent to react, the labelled transition above becomes $a \xrightarrow{-|\bar{a}} \mathbf{0}$. In this case, instead of observing that a can execute an input over a, we observe that it can interact with a context offering an output over a.

So, by using this idea of having contexts as labels, we can derive an LTS from a reactive system. The most immediate way to obtain it is plugging a term P into some context $C[-]$ and observe if a reduction occurs. In this case we have that $P \xrightarrow{C[-]}$. Categorically speaking, this means that $P; C[-]$ matches $l; d$ for some rule $\langle l, r\rangle \in \mathfrak{R}$ and some reactive context d. This situation is formally depicted in Figure 2.4. A commuting diagram like this is called *redex square*.

Definition 2.3 (Saturated transition system)**.** *The* saturated transition system *(*STS*) is defined as follows*

- *states: arrows $P : 0 \to I$ in $\mathbf{C}$, for arbitrary I;*
- *transitions: $P \xrightarrow{C[-]}_{SAT} Q$ if $C[P] \to Q$.*

Note that $C[P]$ is a stand-in for $P; C[-]$: in the rest of the thesis often we will use this notation to allow an easier comparison with the process calculi notation.

Bisimilarity over STS is a congruence and coincides with the definition below.

Definition 2.4 (Saturated bisimulation)**.** *A symmetric relation $\mathcal{R}$ is a* saturated bisimulation *if whenever $P \,\mathcal{R}\, Q$ then $\forall C[-]$*

- *if $C[P] \to P'$ then $C[Q] \to Q'$ and $P' \,\mathcal{R}\, Q'$.*

Saturated bisimilarity $\sim^S$ *is the largest saturated bisimulation.*

$$a \xrightarrow{a} \mathbf{0} \quad \bar{a} \xrightarrow{\bar{a}} \mathbf{0} \quad a \mid \bar{a} \xrightarrow{\tau} \mathbf{0} \quad \frac{P \xrightarrow{\lambda} Q}{P \mid R \xrightarrow{\lambda} Q \mid R} \quad \frac{P \equiv P' \; P' \xrightarrow{\lambda} Q' \; Q' \equiv Q}{P \xrightarrow{\lambda} Q}$$

Figure 2.3: Labelled transition system for the SPC.

$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "{C[-]}"] & & I_3 \arrow[ul, "d"'] \\
 & 0 \arrow[ul, "P"] \arrow[ur, "l"'] &
\end{tikzcd}$$

Figure 2.4: Redex square

Proposition 2.1. $\sim^S$ is the coarsest bisimulation on $\rightarrow$ that is also a congruence.

Note that STS is often infinite-branching since all contexts allowing reductions may occur as labels. Moreover, it has redundant transitions. For example, consider the SPC term a. We have both the transitions $a \overset{-|\bar{a}}{\rightarrow}_{SAT} \mathbf{0}$ and $a \overset{-|\bar{a}|P}{\rightarrow}_{SAT} \mathbf{0} \mid P$, yet P does not "concur" to the reduction. We thus need a notion of "smallest context allowing a reduction". In other words, we have to determine $C[-]$ and d of the redex square in Figure 2.4 in a way such that they are a "least upper bound" of P and l. Categorically, this means to require that the square must be a *pushouts*.

Unfortunately, in many interesting categories of terms, pushouts often do not exist. So, instead of considering the smallest contexts allowing reductions, we consider the "minimal ones" captured by the notion of *idem pushouts*, which exist, unlike pushouts, in many categories of terms. Before giving the definition of idem pushout, we give the definition of relative pushout.

Definition 2.5 (RPO)**.** *Let the diagrams in Figure 2.5 be in a category* $\mathbf{C}$*, and let* (i) *be a commuting diagram. A* candidate *for (i) is any tuple* $\langle I_5, e, f, g\rangle$ *which makes (ii) commute. A* relative pushout (RPO) *is the smallest such candidate, i.e., it satisfies the universal property that given any other candidate* $\langle I_6, e', f', g'\rangle$*, there exists a unique morphism* $h : I_5 \rightarrow I_6$ *such that (iii) and (iv) commute.*

A commuting diagram is called *idem pushout* (IPO) if it has an RPO of a special kind.

Definition 2.6 (IPO)**.** *A commuting square such as diagram (i) of Figure 2.5 is called* idem pushout (IPO) *if* $\langle I_4, c, d, id_{I_4}\rangle$ *is its RPO.*

Hereafter, we say that a reactive system *has redex RPOs* (*IPOs*) if every redex square has an RPO (IPO) as candidate.

IPOs form the basis of the following definition of labelled transition system.

Definition 2.7 (IPO-labelled transition system)**.** *The* IPO-labelled transition system *(*ITS*) is defined as follows*

- *states:* $P : 0 \rightarrow I$ *in* $\mathbf{C}$*, for arbitrary* I*;*
- *transitions:* $P \overset{C[-]}{\rightarrow}_{IPO} r; d$ *if* $d \in \mathbf{D}$*,* $\langle l, r\rangle \in \mathfrak{R}$*, and the redex square of Figure 2.4 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 $C[-]$.

$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "c"] & & I_3 \arrow[ul, "d"'] \\
 & I_1 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "c"] \arrow[r, "e"'] & I_5 \arrow[u, "g"] & I_3 \arrow[ul, "d"'] \arrow[l, "f"] \\
 & I_1 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
 & I_6 & \\
I_2 \arrow[ur, "e'"] \arrow[r, "e"'] & I_5 \arrow[u, "h"] & I_3 \arrow[ul, "f'"'] \arrow[l, "f"]
\end{tikzcd}$$

(iii)

$$\begin{tikzcd}
 & I_4 \\
I_6 \arrow[ur, "g'"] & I_5 \arrow[l, "h"] \arrow[u, "g"']
\end{tikzcd}$$

(iv)

Figure 2.5: RPO

Bisimilarity on ITS is referred to as *IPO-bisimilarity* ($\sim^I$).

In [45], the authors showed that if the reactive system has redex RPOs, then $\sim^I$ is a congruence. To do this they use the composition and decomposition property of IPOs.

Proposition 2.2 (Composition and decomposition)**.** Suppose that diagram (i) of Figure 2.6 has an RPO. Then:

1. (Composition.) if both squares in diagram (ii) of Figure 2.6 are IPOs then so is the exterior rectangle (diagram (iii) of the same figure);
2. (Decomposition.) if the lower square and the exterior one (diagram (iii) of Figure 2.6) of diagram (ii) of Figure 2.6 are IPOs then so is the upper square.

Theorem 2.1. *In a reactive system having redex-RPOs, $\sim^I$ is a congruence.*

From the above theorem and Proposition 2.1, it follows that $\sim^I \subseteq \sim^S$. In [13, 6] the authors show that this inclusion is strict for many formalisms. Moreover, they introduce an efficient characterization of the saturated semantics, called *semi-saturated* semantics. It avoids to consider the whole STS that is usually too big, since it is labelled with all possible contexts allowing reductions, and it uses the ITS, whose labels are just the minimal contexts. So, the abstract semantics is defined in the following way.

Definition 2.8 (Semi-saturated bisimulation)**.** *A symmetric relation $\mathcal{R}$ is a* semi-saturated bisimulation *if whenever $P \,\mathcal{R}\, Q$, then*

- *if $P \xrightarrow{C[-]}_{IPO} P'$ then $C[Q] \rightarrow Q'$ and $P' \,\mathcal{R}\, Q'$.*

Semi-saturated bisimilarity $\sim^{SS}$ *is the largest semi-saturated bisimulation.*

Semi-saturated bisimulations coincide with saturated ones whenever the reactive system has redex IPOs.

Theorem 2.2. *In a reactive system having redex-IPOs, $\sim^{SS} = \sim^S$.*

2.2 The Theory of G-reactive Systems

In several natural examples where process calculi with even simple structural congruences are considered, RPOs either do not exist or do not give the expected equivalence. Therefore, in [64, 63], Sassone and Sobociński proposed an extension of the theory of reactive systems to the 2-categorical setting, in order to consider the structural congruence as an integral part of the theory.

We begin by showing why in a simple calculus with an associative and commutative parallel operator, such as the fragment of the CCS presented in Example 2.1, the application of the theory of RPOs fails.

Example 2.2. Let us consider the reactive system $\mathbb{C}_{SPC}$ shown in Example 2.1 which models a fragment of the CCS, and the two SPC terms $a \mid \bar{a}$ and $b \mid \bar{b}$. It is easy to verify that both terms have just one

$$\begin{tikzcd}
& I_6 & \\
I_2 \arrow[ur, "e;f"] & & I_3 \arrow[ul, "d;g"'] \\
& I_1 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
& & I_6 & & \\
& I_4 \arrow[ur, "f"] & & I_5 \arrow[ul, "g"'] & \\
& & I_2 \arrow[ul, "e"] \arrow[ur, "c"] & & I_3 \arrow[ul, "d"'] \\
& & & I_1 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
& I_6 & \\
I_4 \arrow[ur, "f"] & & I_3 \arrow[ul, "d;g"'] \\
& I_1 \arrow[ul, "a;e"] \arrow[ur, "b"'] &
\end{tikzcd}$$

(iii)

Figure 2.6: IPOs composition and decomposition

$$\begin{tikzcd}
 & 1 & \\
1 \arrow[ur, "-"] \arrow[r, "-"'] & 1 \arrow[u, "-"] & 1 \arrow[ul, "-"'] \arrow[l, "-"] \\
 & 0 \arrow[ul, "a|\bar{a}"] \arrow[ur, "a|\bar{a}"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & 1 & \\
1 \arrow[ur, "-"] \arrow[r, "-"'] & 1 \arrow[u, "-"] & 1 \arrow[ul, "-"'] \arrow[l, "-"] \\
 & 0 \arrow[ul, "b|\bar{b}"] \arrow[ur, "b|\bar{b}"'] &
\end{tikzcd}$$

Figure 2.7: IPOs for the terms $a \mid \bar{a}$ and $b \mid \bar{b}$ (left to right).

IPO transition labelled with the identity contexts $-$, which are shown in Figure 2.7. Therefore, against the intuition, they result to be IPO-bisimilar.

If we indeed consider the standard LTS semantics for the calculus (Figure 2.3), we can distinguish these two terms, since the former has also the two transitions $a \mid \bar{a} \xrightarrow{a} \mathbf{0} \mid \bar{a}$ and $a \mid \bar{a} \xrightarrow{\bar{a}} a \mid \mathbf{0}$, denoting the fact that the term can interact with the environment offering an input, respectively an output, over the channel a. Obviously, the latter term has no transitions with these labels, and so they are not equivalent.

So, going back to the IPO LTS, we would like to derive also the two transitions for the term $a \mid \bar{a}$ corresponding to the ones of the standard LTS semantics shown before. This would mean requiring that also the upper bounds of the two squares of Figure 2.8 are in some sense minimal. Here, to better explain the motivations, we numerate the different occurrences of a to distinguish them, but obviously, it is impossible in our category, where terms are up to structural congruence. It is easy to see that in our category the two squares are not IPOs. In both cases, indeed, the smallest candidate is the quadruple $\langle 1, -, -, - \mid a\rangle$. However, if we could distinguish the different occurrences of a, the upper bound of the left square in Figure 2.8 would be the minimal one where the synchronization uses only the subterm a and the output over the same channel is offered by the context. Similarly, the upper bound of the right square in Figure 2.8 would be the minimal one where the synchronization uses only the subterm $\bar{a}$ and the input is offered by the context.

So, summing up, the fact that in our category, terms are quotiented with respect to the structural congruence makes impossible first to exactly determine which occurrences of a term belong to the redex and consequently to obtain the right LTS.

$$\begin{tikzcd}
 & 1 & \\
1 \arrow[ur, "-|\bar{a}_3"] & & 1 \arrow[ul, "-|\bar{a}_2"'] \\
 & 0 \arrow[ul, "a_1|\bar{a}_2"] \arrow[ur, "a_1|\bar{a}_3"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & 1 & \\
1 \arrow[ur, "-|a_3"] & & 1 \arrow[ul, "-|a_1"'] \\
 & 0 \arrow[ul, "a_1|\bar{a}_2"] \arrow[ur, "a_3|\bar{a}_2"'] &
\end{tikzcd}$$

Figure 2.8: Redex squares for the SPC.

The approach proposed in [64, 63] to solve the problems of the theory of reactive systems discussed above consists in keeping explicitly the derivation of structural congruence between terms by using categories which have a 2-dimensional structure, that is, "arrows between arrows", called *2-cells*.

Before introducing the generalization of the notions of reactive system, RPO and IPO to the setting of 2-categories, we shortly recall the definition of 2-categories.

Definition 2.9 (2-category). *A 2-category* $\mathbf{C}$ *is a category consisting of*

1. *a class of objects* $X, Y, Z, \ldots$

2. *for any* $X, Y \in \mathbf{C}$*, a category* $\mathbf{C}(X, Y)$*. The objects of* $\mathbf{C}(X, Y)$ *are arrows, and they are called* 1-cells*, or simply arrows, and denoted by* $f : X \to Y$*. Identity arrows are instead denoted by*

$id_X : X \to X$. The morphisms of $\mathbf{C}(X,Y)$ are called 2-cells*. They are denoted by $\alpha : f \Rightarrow g$ and represented as in Figure 2.9. Composition in $\mathbf{C}(X,Y)$ is referred to as "vertical" composition and it is denoted by $\bullet$. Identity 2-cells are denoted by $1_f : f \Rightarrow f$.*

3. *for each $X, Y, Z \in \mathbf{C}$ a functor $* : \mathbf{C}(Y,Z) \times \mathbf{C}(X,Y) \longrightarrow \mathbf{C}(X,Z)$, called "horizontal" composition, which is associative and admits 1_{id_X} as identities.*

The role of 2-cells in the approach proposed in [64, 63] is to represent structural congruences. This means that if there exists a 2-cell $\alpha : f \Rightarrow g$, then f and g represent two terms structurally equivalent, and α is a proof of this equivalence. They therefore consider 2-categories whose 2-cells are isomorphisms. Since the categories whose morphisms are all isomorphisms are commonly known as groupoids, these 2-categories are precisely the groupoid-enriched categories, or G-categories.

Definition 2.10 (G-Category)**.** *A G-category is a 2-category whose 2-cells are all isomorphisms.*

Now, we can introduce the generalization of the notion of reactive system to the setting of G-categories.

Definition 2.11 (G-reactive System)**.** *A G-reactive system $\mathbb{C}$ consists of*

1. *a G-category $\mathbf{C}$;*
2. *a distinguished object $0 \in \mathbf{C}$;*
3. *a set $\mathbf{D} \subseteq \mathbf{C}$ of 2-cells closed, composition-reflecting* 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* reduction rules.

The closure property means that given $d \in \mathbf{D}$ and $\alpha : d \Rightarrow d'$ in $\mathbf{C}$ then $d' \in \mathbf{D}$.

Definition 2.12 (Reduction Relation)**.** *Given a G-reactive system $\langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$, the reduction relation $\to$ is defined as follows: $P \to Q$ if there exist 2-cells $P \Rightarrow l; d$ and $r; d \Rightarrow Q$ for some reduction rule $\langle l, r \rangle \in \mathfrak{R}$ and a reactive context $d \in \mathbf{D}$ (Figure 2.10).*

In the following, we present a G-reactive systems modeling SPC .

Example 2.3. Let us consider again the fragment of Milner's CCS introduced in Example 2.1. The corresponding G-reactive system $\mathbb{C}_{SPC} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ is defined as follows.

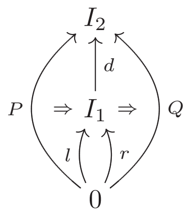
The 2-category $\mathbf{C}$ has the same objects and arrows of the category underlying the reactive system defined in Example 2.1. The only difference is that here terms are considered quotiented only by the associativity equation (the left rule in the second row of Figure 2.2). Intuitively, here arrows could be seen as sequences where the order of the elements is important, while in Example 2.1 as multisets.

Isomorphic 2-cells between terms intuitively correspond to the commutativity axiom of the structural congruence (the right rule in the second row of Figure 2.2). So, a 2-cell between two terms is a permutation which swaps parallel components (where by component we mean an occurrence of an input/output on a channel or a hole). So, an arrow representing a term composed of n parallel components is the source of $n!$ 2-cells determined by the permutations of its components in parallel. Thus, for instance, there are two automorphisms on $a \mid a : 0 \to 1$, the identity, and the automorphism which swaps the two copies of a.

The distinguished object, the reactive contexts and the reduction rules are defined as in Example 2.1.

$$\begin{tikzcd} X \arrow[r, bend left, "f"{name=U}] \arrow[r, bend right, "g"'{name=D}] & Y \arrow[from=U, to=D, Rightarrow, "\alpha"] \end{tikzcd}$$

Figure 2.9: The 2-cell $\alpha : f \Rightarrow g$.



$$\begin{tikzcd}
 & I_2 & \\
 & I_1 \arrow[u, "d"] & \\
 & 0 \arrow[uu, bend left=60, "P"] \arrow[uu, bend right=60, "Q"'] \arrow[u, bend left, "l"] \arrow[u, bend right, "r"'] &
\end{tikzcd}$$

Figure 2.10: Reduction relation for G-reactive systems

In this setting, a redex square is a diagram as the one in Figure 2.11, where there exists an explicit isomorphism α between the terms obtained by plugging P into the context $C[-]$ and the redex l into the reactive context d.

In the following, we present a generalization of the notion of RPO to G-categories. This notion is used to formalize the idea of the "smallest" context allowing a reduction in a G-reactive system. We refer the reader to [61] for a more detailed presentation.

Definition 2.13 (GRPO)**.** *Let the diagrams in Figure 2.12 be in a G-category* $\mathbf{C}$*. A candidate for the diagram (i) is any tuple* $\langle I_5, n, o, p, \beta, \gamma, \delta\rangle$ *such that* $(1_h * \gamma) \bullet (\beta * 1_p) \bullet (1_g * \delta) = \alpha$*. This means that the 2-cells* γ, β, δ*, illustrated in diagram (ii), paste together to give* α*. A* groupoidal-relative-pushout *(GRPO) is a candidate which satisfies the universal property, i.e., for any other candidate* $\langle I_6, n', o', p', \beta', \gamma', \delta'\rangle$ *there exists a* mediating morphism*, that is, a quadruple* $\langle q : I_5 \to I_6, \varphi : n' \Rightarrow n; q, \psi : o; q \Rightarrow o', \tau : q; p' \Rightarrow p\rangle$ *illustrated in diagrams (iii) and (iv). The equations that need to be satisfied are:* $1)\gamma' \bullet (\varphi * 1_{p'}) \bullet (1_n * \tau) = \gamma$*;* $2)(1_o * \tau^{-1}) \bullet (\psi * 1_{p'}) \bullet \delta' = \delta$*;* $3)(1_h * \varphi) \bullet (\beta * 1_q) \bullet (1_g * \psi) = \beta'$*. Such a mediating morphism must be essentially unique, namely, for any other mediating morphism* $\langle q', \varphi', \psi', \tau'\rangle$ *there must exist a unique 2-cell* $\xi : q \Rightarrow q'$ *which makes the two mediating morphisms compatible, i.e.:* $1)\varphi \bullet (1_n * \xi) = \varphi'$*;* $2)(1_o * \xi^{-1}) \bullet \psi = \psi'$*;* $3)(\xi * 1_{p'}) \bullet \tau' = \tau$*.*

Diagram (i) of Figure 2.12 is a GIPO if it has a special kind of GRPO.

Definition 2.14 (GIPO)**.** *A square such as diagram (i) of Figure 2.12 is called G-idem pushout (GIPO) if* $\langle I_4, f, m, id_{I_4}, \alpha, 1_f, 1_m\rangle$ *is its GRPO.*

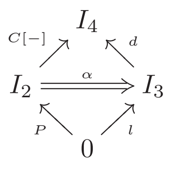
Analogously to reactive systems, we say that a G-reactive system has redex GRPOs (GIPOs) if every redex square has a GRPO (GIPO) as candidate.

In the following we define an LTS by using the notion of GIPO.

Definition 2.15 (GIPO-labelled transition system)**.** *Let* $\mathbb{C}$ *be a G-reactive system and let* $\mathbf{C}$ *be its underlying 2-category. The* GIPO-labelled transition system $\text{GLTS}(\mathbb{C})$ *is defined as follows*

- *states:* $P : 0 \to I$ *in* $\mathbf{C}$*, for arbitrary* I*;*
- *transitions:* $P \xrightarrow{C[-]}_{GIPO} P'$ *if there exists* $d \in \mathbf{D}$*,* $\langle l, r\rangle \in \mathfrak{R}$*, and a 2-cell* $\alpha : P; C[-] \Rightarrow l; d$ *such that the redex square in Figure 2.11 is a GIPO and* P' *is isomorphic to* $r; d$*.*

Bisimilarity on $\text{GLTS}(\mathbb{C})$ *is referred to as GIPO-bisimilarity (*$\sim^{GIPO}$*).*



$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "{C[-]}"] \arrow[rr, Rightarrow, "\alpha"] & & I_3 \arrow[ul, "d"'] \\
 & 0 \arrow[ul, "P"] \arrow[ur, "l"'] &
\end{tikzcd}$$

Figure 2.11: Redex square in G-reactive systems

$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "f"] \arrow[rr, Rightarrow, "\alpha"] & & I_3 \arrow[ul, "m"'] \\
 & I_1 \arrow[ul, "h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
 & I_4 & \\
I_2 \arrow[ur, "f"] \arrow[r, "n"] \arrow[ur, phantom, shift right=3ex, "\gamma \Rightarrow"] & I_5 \arrow[u, "p"] \arrow[ur, phantom, shift right=3ex, "\delta \Rightarrow"] & I_3 \arrow[ul, "m"'] \arrow[l, "o"'] \\
 & I_1 \arrow[ul, "h"] \arrow[ur, "g"'] \arrow[u, phantom, "\overset{\beta}{\Rightarrow}"] &
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
 & I_6 & \\
I_2 \arrow[ur, "n'"] \arrow[r, "n"] \arrow[ur, phantom, shift right=3ex, "\varphi \Rightarrow"] & I_5 \arrow[u, "q"] \arrow[ur, phantom, shift right=3ex, "\psi \Rightarrow"] & I_3 \arrow[ul, "o'"'] \arrow[l, "o"']
\end{tikzcd}$$

(iii)

$$\begin{tikzcd}
 & I_4 \\
I_6 \arrow[ur, "p'"] \arrow[ur, phantom, shift right=2ex, "\tau \Rightarrow"] & I_5 \arrow[u, "p"'] \arrow[l, "q"]
\end{tikzcd}$$

(iv)

Figure 2.12: GRPO

Example 2.4. Let us consider the G-reactive system $\mathbb{C}_{SPC}$ previously defined in Example 2.3. It is easy to verify that in its underlying 2-category, the two diagrams in Figure 2.8 are GIPOs, by considering for the left square the 2-cell $\alpha : a_1 \mid \bar{a}_2 \mid \bar{a}_3 \Rightarrow a_1 \mid \bar{a}_3 \mid \bar{a}_2$, which swaps the two outputs over a, and for the right square the 2-cell $\alpha' : a_1 \mid \bar{a}_2 \mid a_3 \Rightarrow a_3 \mid \bar{a}_2 \mid a_1$, which swaps the two inputs over a.

Theorem 2.3. *In a reactive system having redex-GRPOs, $\sim^{GIPO}$ is a congruence.*

2.3 Graph Transformation and the Borrowed Context Technique

In previous sections, we presented the theory of (G-)reactive systems aimed at deriving behavioral congruences for those specification formalisms whose operational semantics is provided by unlabelled rewriting rules. The *borrowed contexts* (BCs) technique, developed by Ehrig and König [27, 28], offers a solution to the same problem in the double-pushout (DPO) approach to graph rewriting.

In the following, we first introduce the BC mechanism (Section 2.3.1) and then we briefly show the relationship with the theory of G-reactive systems.

2.3.1 DPO Rewriting with Borrowed Contexts

This section introduces *double-pushout* (DPO) rewriting and its interactive extension with *borrowed contexts* (BCs) [27, 28]. We present them by relying on adhesive categories as in [64]. Adhesive categories were introduced by Lack and Sobocinski in [44]. They are categories in which pushouts along monomorphisms are well-behaved. Various graphical structures used in computer science form adhesive categories. Some examples are directed graphs, typed graphs and hypergraphs.

Below we recall the definition of adhesive categories.

Definition 2.16 (Adhesive Categories). *A category is called* adhesive *if*

- *it has pushouts along monos;*
- *it has pullbacks;*
- *pushouts along monos are* Van Kampen *(*VK*) squares.*

Referring to Figure 2.13, a VK *square is a pushout like (i), such that for each commuting cube as in (ii) having (i) as bottom face and the back faces of which are pullbacks, the front faces are pullbacks if and only if the top face is a pushout.*

As shown in [64], adhesive categories provide an elegant setting in which one can develop the well-known theory of double-pushout graph rewriting [29, 26].

In order to uniformly introduce DPO and BCs, we consider DPO derivations for *systems with interface*: morphisms $J \to G$ where G represents a system and J its interface.

Definition 2.17 (Production). *Let* $\mathbf{A}$ *an adhesive category. A* production *or* rewrite rule $p : (L \leftarrowtail I \to R)$ *is a a production name* p *and a span* $L \leftarrowtail I \to R$ *in* $\mathbf{A}$*, where the left-hand side* $I \rightarrowtail L$ *is monic.*

$$\begin{tikzcd}
 & C \arrow[dl] \arrow[dr] & \\
A \arrow[dr] & & B \arrow[dl] \\
 & D &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & C' \arrow[dll] \arrow[dr] \arrow[ddd] & \\
A' \arrow[dr] \arrow[ddd] & & & B' \arrow[dll] \arrow[ddd] \\
 & D' \arrow[ddd] & & \\
 & & C \arrow[dll] \arrow[dr] & \\
A \arrow[dr] & & & B \arrow[dll] \\
 & D & &
\end{tikzcd}$$

(i) (ii)

Figure 2.13: A pushout square (i), and a commutative cube (ii).

Definition 2.18 (DPO adhesive rewriting system)**.** *A* DPO adhesive rewriting system *(ARS) is a pair $\langle \mathbf{A}, P\rangle$, where $\mathbf{A}$ is an adhesive category and P is a set of productions with different names.*

In the definitions below, we refer to a chosen ARS $S = \langle \mathbf{A}, P\rangle$.

Definition 2.19 (DPO derivation for systems with interfaces)**.** *Let $J \to G$ and $J \to H$ be two systems with interface and $p : (L \leftarrowtail I \to R)$ a production. A* match *of p in G is a morphism $m : L \to G$. A* direct derivation *from $J \to G$ to $J \to H$ via p and m is a commuting diagram as depicted below, where (1) and (2) are pushouts and the bottom triangles commute. In this case we write $J \to G \Longrightarrow J \to H$.*

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & I \arrow[l, tail] \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\
G & C \arrow[l, tail] \arrow[r] & H \\
 & J \arrow[ul] \arrow[u, "k"] \arrow[ur] &
\end{tikzcd}$$

The morphism $k : J \to C$ (making the left triangle commute) is unique, whenever it exists. If such a morphism does not exist, the rewriting step is not feasible. Note that the standard DPO derivations can be seen as a special instance of these, obtained considering as interface J the empty graph.

In these derivations, the left-hand side L of a production must then occur completely in G. In a BC derivation L might occur partially in G, since the latter may interact with the environment through the interface J in order to exactly match L. Those BCs are the "smallest" contexts needed to obtain the image of L in G, and they may be used as suitable labels. Given an ARS S, BC(S) denotes the LTS derived via the BC mechanism defined below.

Definition 2.20 (Rewriting with borrowed contexts)**.** *Given a production $p : L \leftarrowtail I \to R$, a system with interface $J \to G$ and a span of monos $d : G \leftarrowtail D \rightarrowtail L$, we say that $J \to G$* reduces to *$K \to H$ with label $J \rightarrowtail F \leftarrow K$* via *$p$ and d if there are objects G^+, C and additional morphisms such that the diagram below commutes and the squares are either pushouts (PO) or pullbacks (PB). We write $J \to G \xrightarrow{J \rightarrowtail F \leftarrow K} K \to H$, called* rewriting step with borrowed context.

$$\begin{tikzcd}
D \arrow[r, tail] \arrow[d, tail] \arrow[dr, phantom, "PO"] & L \arrow[d, tail] \arrow[dr, phantom, "PO"] & I \arrow[l, tail] \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "PO"] & R \arrow[d, tail] \\
G \arrow[r, tail] & G^+ & C \arrow[l, tail] \arrow[r] & H \\
J \arrow[r, tail] \arrow[u] \arrow[ur, phantom, "PO"] & F \arrow[u] \arrow[ur, phantom, "PB"] & K \arrow[l] \arrow[u] \arrow[ur, dotted] &
\end{tikzcd}$$

Consider the diagram above. The upper left-hand square merges the left-hand side L and the object G to be rewritten according to a partial match $G \leftarrowtail D \rightarrowtail L$. The resulting G^+ contains a total match of L and is rewritten as in the DPO approach, producing the two other squares in the upper row. The pushout in the lower row gives the borrowed (or minimal) context F which is missing for obtaining a total match of L, along with a morphism $J \rightarrowtail F$ indicating how F should be pasted to G. Finally, the interface for H is obtained by "intersecting" F and C via a pullback.

The two pushout complements that are needed in Definition 2.20, namely C and F, may not exist. In this case, the rewriting step is not feasible.

Note that some morphisms that in the diagram of Definition 2.19 can be arbitrary, in the diagram of Definition 2.20 are instead required to be mono. This is necessary in order to obtain a bisimilarity over the derived LTS which is a congruence.

2.3.2 Relating Borrowed Contexts and G-Reactive Systems

We are now ready for showing that adhesive rewriting systems are instances of G-reactive systems, as previously proved in [63]. We consider cospans as contexts, and for this reason we need to work in bicategories [5] (with iso 2-cells) instead of G-categories. For our aim it is enough to know that a bicategory can be described, roughly, as a 2-category where associative and identity laws of composition hold up to isomorphism. In order to transfer the notions of GIPOs and GRPOs (in Section 2.2) to bicategories, it suffices to introduce the coherent associativity isomorphisms where necessary.

Bicategories of Cospans. Let $\mathbf{A}$ be an adhesive category with chosen pushouts. This means that for each span $A \leftarrow B \rightarrow C$, there exists a unique chosen cospan $A \rightarrow A +_B C \leftarrow C$ such that the resulting square is a pushout. The bicategory of cospans of $\mathbf{A}$ has the same objects as $\mathbf{A}$ and morphism pairs $I_1 \xrightarrow{i_C} C \xleftarrow{o_C} I_2$ as arrows from I_1 to I_2, denoted $C^{o_C}_{i_C} : I_1 \rightarrow I_2$. Objects I_1 and I_2 are thought of as the input and the output interface of $C^{i_C}_{o_C}$.

Given the cospans $C^{o_C}_{i_C} : I_1 \rightarrow I_2$ and $D^{o_D}_{i_D} : I_2 \rightarrow I_3$, their composition $C^{o_C}_{i_C}; D^{o_D}_{i_D} : I_1 \rightarrow I_3$ is the cospan obtained by taking the chosen pushout of o_C and i_D, as depicted in Figure 2.14. Note that, since arrows composition is a chosen pushout, it is associative only up to isomorphism. This is the reason why cospans form a bicategory and not a 2-category.

A 2-cell $h : C^{o_C}_{i_C} \Rightarrow D^{o_D}_{i_D} : I_1 \rightarrow I_2$ is an arrow $h : C \rightarrow D$ in $\mathbf{A}$ satisfying $i_C; h = i_D$ and $o_C; h = o_D$, and it is *isomorphic* if h is an isomorphism in $\mathbf{A}$.

A cospan $C^{o_C}_{i_C}$ is *input linear* when i_C is mono in $\mathbf{A}$, and the composition of two input linear cospans yields another input linear cospan. For this reason, we can define the *input linear cospans bicategory* over $\mathbf{A}$, denoted by $ILC(\mathbf{A})$, as the bicategory consisting of input linear cospans and isomorphic 2-cells.

From adhesive rewriting systems to G-reactive systems. Consider an ARS $S = \langle \mathbf{A}, P \rangle$, where the adhesive category $\mathbf{A}$ has initial object 0. This can be seen as a G-reactive system where

- the base category is $ILC(\mathbf{A})$,
- the distinguished object is 0 (the initial object),
- all arrows in $ILC(\mathbf{A})$ are reactive,
- rules are pairs $\langle 0 \rightarrow L \leftarrowtail I, 0 \rightarrow R \leftarrow I \rangle$ for any $L \leftarrowtail I \rightarrow R$ rule in P .

For an ARS S, $\mathbb{C}_S$ denotes its associated G-reactive system. A system with interface $J \rightarrow G$ in S can be thought as the arrow $0 \rightarrow G \leftarrow J$ of $ILC(\mathbf{A})$.

$$\begin{tikzcd}
 & & C +_{I_2} D & & \\
I_1 \arrow[r, "i_C"'] & C \arrow[ur] & I_2 \arrow[l, "o_C"] \arrow[r, "i_D"'] \arrow[u, phantom, "PO"] & D \arrow[ul] & I_3 \arrow[l, "o_D"]
\end{tikzcd}$$

Figure 2.14: Cospans composition.

The translation presented above preserves semantics.

Proposition 2.3 ([33]). $(J \to G) \Longrightarrow (J \to H)$ in S iff $(J \to G) \to (J \to H)$ in $\mathbb{C}_S$.

The above result allows for stating the correspondence between ARSs and BCs: GIPOs for G-reactive systems over input linear cospans are equivalent to BCs for ARSs.

Proposition 2.4 ([63]). $\textsc{bc}(S) = \textsc{glts}(\mathbb{C}_S)$.

Chapter 3

Graphical encodings for mobile ambients and asynchronous CCS

This chapter present two graphical encodings for the mobile ambients calculus [16] and the asynchronous CCS [31], respectively. In particular, the graphical encoding of the asynchronous CCS is basically an adaptation of the one for the synchronous version in [7]. We instead present a new encoding for the mobile ambients calculus and we provide an in-depth study of its features. Both encodings will be useful for the examples we will use to illustrate the adequacy of the results presented in Chapters 5 and 6.

For both calculi, the proposed encodings use unstructured (i.e., non-hierarchical) graphs and they are sound and complete with respect to the structural congruence of the corresponding calculus (i.e., two processes are equivalent if and only if mapped to isomorphic graphs). As far as the mobile ambients calculus is concerned, with respect to alternative proposals for the graphical implementation of the calculus, our encoding exploits the dichotomy between the tree structure of a process and the topology associated to its activation points, i.e., to those ambients that actually allow for the evolution of the subprocesses they contain. In the encoding for the asynchronous CCS this is not necessary, because the syntactical and the activation dependence between the operators of a process exactly coincide. For both calculi the encoding is then exploited to recast the operational semantics of the calculus by an easy and natural presentation via DPO rules, thus inheriting the wealth of tools and techniques for system analysis that are available for graph transformation. Moreover, in the case of mobile ambients our solution faithfully captures a basic feature of the calculus: ambients can be nested and reductions are propagated across the nesting.

The adoption of graph transformation for simulating the reduction semantics of process calculi allowed for some technology transfer. One of its foremost applications has been the distillation of observational semantics for such calculi, by relying on the borrowed context mechanism: an application of this methodology to mobile ambients and asynchronous CCS will be shown in Chapter 4. In this chapter, instead, we profitably exploit another feature of the graph transformation formalism, namely, the possibility of defining suitable concurrent semantics. This allows for obtaining such a semantics also for any encoded calculus, hence offering a better understanding of process behaviour. In particular, we exploit the information about dependencies among (causally related) rewriting steps offered by the concurrent semantics of mobile ambients to identify *interferences* between process reductions, formalising the taxonomy proposed in [46].

This chapter is organised as follows. Section 3.1 briefly recalls the calculus of mobile ambients. In Section 3.2 we present the main definitions concerning (typed hyper-)graphs and their extension with interfaces, while Section 3.3 recalls the DPO approach to their rewriting. Then, in Section 3.4 we introduce a graphical encoding for processes of the mobile ambients calculus, and we present our first result, namely, that our encoding is sound and complete with respect to a slight variant of the structural congruence of mobile ambients. The main result of our work is presented in Section 3.5, which introduces a graph transformation system for modelling the reduction semantics of mobile ambients. In Section 3.6 instead we propose a graph transformation system to recover a normal form for each graphical encoding of a process. This allows us to recast the standard structural congruence of mobile ambients in terms of graph isomorphism. Section 3.7 discusses the concurrency features of our graph transformation system for mobile ambients, and shows how the notion of independence between rewriting steps may be used for giving a formal definition of both *plain* and *grave interferences* among process reductions, as introduced in [46]. Then, Section 3.8 discusses related work for the encoding of mobile ambients, while Section 3.9 briefly introduces the asynchronous CCS, a graphical encoding for it, and a graph transformation system modelling its reduction semantics.

3.1 Mobile Ambients

In this section we (very) briefly recall the calculus of mobile ambients [16]. In particular, we introduce the syntax and the reduction semantics for its (finite) fragment without the communication primitives.

Figure 3.1 shows the syntax of the calculus. We assume a set $\mathcal{N}$ of *names* ranging over by $m, n, o, \ldots$. Also, we let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of processes.

The restriction operator $(\nu n)P$ binds n in P. A name n occurring in the scope of the operator (νn) is called *bound*, otherwise it is called *free*. We denote the set of free names of a process P by $fn(P)$. We adopt the standard notion of α-conversion of bound names and the standard definition for name substitution. We write $P\{m/n\}$ for the process obtained by replacing each free occurrence of n in P with m, and by α-converting the bound names to avoid conflicts with m.

The semantics of the mobile ambients calculus is given by the combination of an equivalence between processes and a pre-order relation among them. The *structural congruence*, denoted by $\equiv$, is the least relation on processes that satisfies the equations and the rules shown in Figure 3.2. The congruence relates processes which intuitively specify the same system, up-to a syntactical rearrangement of its components, and it is used to define the operational semantics.

The *reduction relation*, denoted by $\rightarrow$, describes the evolution of processes over time: $P \rightarrow Q$ means that P reduces to Q, that is, P can execute a computational step and it is transformed into Q. Figure 3.3 shows the reduction rules. The first three rules are the only three axioms for the reduction relation. In particular, the *Red-In* rule enables an ambient n to enter a sibling ambient m. The *Red-Out* rule enables an ambient n to get out of its parent ambient m. Finally, the last axiom allows to dissolve the boundary of an ambient n. The *Red-Res*, *Red-Amb* and *Red-Par* rules say that a reduction can occur underneath restriction, ambient and parallel composition, respectively. Finally, the last rule says that the reduction relation is closed under the structural congruence $\equiv$.

3.1.1 An alternative congruence

As we stated above, the structural congruence is pivotal in the definition of the reduction relation. It is possible to take into account different structural congruence relations. We denote by $\equiv'$ the least relation that satisfies also the equation in Figure 3.4 besides those in Figure 3.2, and by $\rightarrow'$ the reduction relation defined by the rules shown in Figure 3.3, but closed under the structural congruence $\equiv'$.

Note that considering the structural congruence $\equiv'$ does not change substantially the reduction semantics. Indeed, the mapping from abstract processes according to $\equiv$ into abstract processes according to $\equiv'$

$P, Q ::=$	processes
$\mathbf{0}$	inactivity
$n[P]$	ambient
$M.P$	action
$(\nu n)P$	restriction
$P_1 \mid P_2$	composition
$M ::=$	capabilities
$in\ n$	can enter n
$out\ n$	can exit n
$open\ n$	can open n

Figure 3.1: Syntax of mobile ambients.

$P \mid Q \equiv Q \mid P$	(Cong-Par-Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Cong-Par-Ass)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Cong-Res-Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad$ if $n \notin fn(P)$	(Cong-Res-Par)
$(\nu n)m[P] \equiv m[(\nu n)P] \quad$ if $n \neq m$	(Cong-Res-Amb)
$P \mid \mathbf{0} \equiv P$	(Cong-Zero-Par)
$(\nu n)P \equiv (\nu m)(P\{m/n\})$ if $m \notin fn(P)$	(Cong-α)

Figure 3.2: Axioms of the structural congruence without the axiom $(\nu n)\mathbf{0} \equiv \mathbf{0}$.

$n[in\ m.P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R]$	(Red-In)
$m[n[out\ m.P \mid Q] \mid R] \to n[P \mid Q] \mid m[R]$	(Red-Out)
$open\ n.P \mid n[Q] \to P \mid Q$	(Red-Open)
$P \to Q \Rightarrow (\nu n)P \to (\nu n)Q$	(Red-Res)
$P \to Q \Rightarrow n[P] \to n[Q]$	(Red-Amb)
$P \to Q \Rightarrow P \mid R \to Q \mid R$	(Red-Par)
$(P' \equiv P, P' \to Q', Q' \equiv Q) \Rightarrow P \to Q$	(Red-Cong)

Figure 3.3: Reduction relation.

$(\nu n)\mathbf{0} \equiv' \mathbf{0}$	(Cong-Zero-Res)

Figure 3.4: The additional axiom of the structural congruence $\equiv'$.

faithfully preserves the reduction semantics, as discussed in [36] and stated by next proposition.

Proposition 3.1. Let P, Q be processes. If $P \rightarrow Q$, then $P \rightarrow' Q$. Vice versa, if $P \rightarrow' Q$, then there exists a process R, such that $P \rightarrow R$ and $Q \equiv' R$.

3.2 Graphs and Graphs with interfaces

This section presents some definitions concerning (hyper-)graphs, typed graphs and graphs with interfaces. It also introduces two operators on graphs with interfaces. We refer to [15] and [19] for a detailed introduction.

Definition 3.1 (Graphs)**.** *A (hyper-)graph is a quadruple $\langle V, E, s, t\rangle$ where V is the set of nodes, E is the set of edges and $s, t : E \rightarrow V^*$ are the source and target functions.*

From now on we denote the components of a graph G by V_G, E_G, s_G and t_G.

Definition 3.2 ((Partial) graph morphisms)**.** *Let G, G' be graphs. A (hyper-)partial graph morphism $f : G \rightarrow G'$ is a pair of partial functions $\langle f_V, f_E\rangle$ such that $f_V : V_G \rightarrow V_{G'}$, $f_E : E_G \rightarrow E_{G'}$ and they preserve the source and target functions, i.e., if $f_E(e)$ is defined, then $(f_V)^*(s_G(e)) = s_{G'}(f_E(e))$ and $(f_V)^*(t_G(e)) = t_{G'}(f_E(e))$. We say that the graph morphism is total if f_V and f_E are so.*

In the following, if not differently specified, morphisms will be total.

The category of graphs and total morphisms is denoted by $\mathbf{Graph}$. We now give the definition of typed graph [20], i.e., a graph labelled over a structure that is itself a graph.

Definition 3.3 (Typed graphs)**.** *Let T be a graph. A* typed graph *G over T is a graph $|G|$ with a graph morphism $\tau_G : |G| \rightarrow T$.*

Definition 3.4 (Typed graph morphisms)**.** *Let G, G' be typed graphs over T. A* typed graph morphism *$f : G \rightarrow G'$ is a graph morphism $f : |G| \rightarrow |G'|$ consistent with the typing, i.e., such that $\tau_G = \tau_{G'} \circ f$.*

The category of graphs typed over T is denoted by T-$\mathbf{Graph}$. In the following, we assume a chosen type graph T.

To define the encoding for processes inductively, we need operations to compose graphs. So, we equip typed graphs with suitable "handles" for interacting with an environment. The following definition introduces graphs with interfaces.

Definition 3.5 (Graphs with interfaces)**.** *Let J, K be typed graphs. A graph with input interface J and output interface K is a triple $\mathbb{G} = \langle j, G, k\rangle$, where G is a typed graph, $j : J \rightarrow G$ and $k : K \rightarrow G$ are injective typed graph morphisms, and they are called input and output morphisms, respectively.*

Definition 3.6 (Interface graph morphisms)**.** *Let $\mathbb{G}, \mathbb{G}'$ be graphs with the same interface. An interface graph morphism $f : \mathbb{G} \Rightarrow \mathbb{G}'$ is a typed graph morphism $f : G \rightarrow G'$ between the underlying typed graphs that preserves the input and output morphisms.*

We denote by $J \xrightarrow{j} G \xleftarrow{k} K$ a graph with input interface J and output interface K. If the interfaces J and K are *discrete*, i.e., they contain only nodes, we represent them by sets. With an abuse of notation, in the following we refer to the nodes belonging to the image of the input (output) morphism as inputs (outputs, respectively). We often refer implicitly to a graph with interfaces as the representative of its isomorphism class. Moreover, we sometimes denote the class of isomorphic graphs and its components by the same symbol.

3.2.1 Two operations on graphs

Now, we define two binary operators on graphs with discrete interfaces.

Definition 3.7 (Sequential composition)**.** *Let $\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$ and $\mathbb{G}' = K \xrightarrow{j'} G' \xleftarrow{k'} I$ be graphs with discrete interfaces. Their sequential composition is the graph with discrete interfaces $\mathbb{G} \circ \mathbb{G}' = J \xrightarrow{j''} G'' \xleftarrow{k''} I$, where G'' is the disjoint union $G \uplus G'$, modulo the equivalence on nodes induced by $k(x) = j'(x)$ for all $x \in V_K$ and with the obvious source and target functions, and j'' and k'' are the uniquely induced arrows.*

Before defining the parallel composition between graphs with interfaces, we introduce the definition of compatible graphs.

Definition 3.8 (Compatible graphs)**.** *Let $\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$ and $\mathbb{G}' = J' \xrightarrow{j'} G' \xleftarrow{k'} K'$ be graphs with discrete interfaces. We say that $\mathbb{G}$ and $\mathbb{G}'$ are* compatible *if $\tau_J(x) = \tau_{J'}(x)$ for all $x \in V_J \cap V_{J'}$ and $\tau_K(y) = \tau_{K'}(y)$ for all $y \in V_K \cap V_{K'}$.*

Definition 3.9 (Parallel composition)**.** *Let $\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$ and $\mathbb{G}' = J' \xrightarrow{j'} G' \xleftarrow{k'} K'$ be compatible graphs with discrete interfaces. Their parallel composition is the graph with discrete interfaces $\mathbb{G} \otimes \mathbb{G}' = (J \cup J') \xrightarrow{j''} G'' \xleftarrow{k''} (K \cup K')$, where G'' is the disjoint union $G \uplus G'$, modulo the equivalence on nodes induced by $j(x) = j'(x)$ for all $x \in V_J \cap V_{J'}$ and $k(y) = k'(y)$ for all $y \in V_K \cap V_{K'}$ and with the obvious source and target functions, and j'', k'' are the uniquely induced arrows.*

Intuitively, the sequential composition $\mathbb{G} \circ \mathbb{G}'$ is obtained by taking the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{G}'$, and gluing the outputs of $\mathbb{G}$ with the corresponding inputs of $\mathbb{G}'$. Similarly, the parallel composition $\mathbb{G} \otimes \mathbb{G}'$ is obtained by taking the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{G}'$, and gluing the inputs (outputs) of $\mathbb{G}$ with the corresponding inputs (outputs) of $\mathbb{G}'$. Note that both operations are defined on "concrete" graphs. However, their results do not depend on the choice of the representatives of their isomorphism classes.

Definition 3.10 (Graph expression)**.** *A* graph expression *is a term over the syntax containing all graphs with discrete interfaces as constants, and parallel and sequential composition as binary operators. We say that an expression is* well-formed *if all the occurrences of both sequential and parallel composition are defined for the interfaces of their arguments, according to Definitions 3.7 and 3.9.*

The interfaces of a well-formed graph expression are computed inductively from the interfaces of the graphs occurring in it; the value of the expression is the graph obtained by evaluating all its operators.

3.2.2 Applying the operations

Let us consider the graphs with interfaces $\mathbb{G}_{amb} = \{a, p\} \to G_{amb} \leftarrow \{a, p, n\}$ and $\mathbb{G}_{in} = \{a, p, n\} \to G_{in} \leftarrow \{m\}$ depicted in Figure 3.5. The graph on the left is just composed of the hyper edge amb, which has two source nodes, one of type $\diamond$ and another one of type $\bullet$, while the two target nodes are respectively of type $\circ$ and $\bullet$. The source nodes are in the input interface, while the output interface is composed of the target nodes plus the source node $\diamond$. Note that the dotted arrows represent input and output morphisms. The graph on the right is instead composed of a node $\circ$ plus the hyper edge in, which has two source nodes respectively of type $\diamond$ and $\bullet$, and three target nodes respectively of type $\circ$, $\bullet$ and $\diamond$. The input interface is composed of the source nodes of the in edge plus the isolated $\circ$ node, while the output interface, besides of this last node, contains the target nodes of in.

As we will see later, the two graphs above respectively represent the graphical operator modelling an ambient n, and the graphical encoding for the process $in\ m.\mathbf{0}$, plus an isolated node $\circ$.

Since the output interface of the graph $\mathbb{G}_{amb}$ coincides with the input interface of $\mathbb{G}_{in}$, we can compute their sequential composition, which results in the graph with interfaces shown on the left of Figure 3.6. It is obtained by the disjoint union of G_{amb} and G_{in}, gluing the nodes of the former that are in the output interface with the nodes of the latter that are in the input interface. Moreover, as it will become clearer later, the graph obtained by the sequential composition represents the graphical encoding of the process $P = n[in\ m.\mathbf{0}]$.

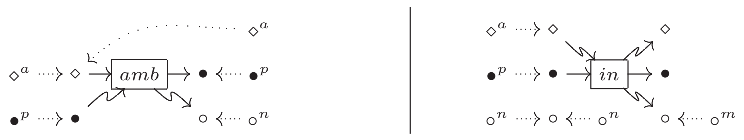


Figure 3.5: Graphs with interfaces $\mathbb{G}_{amb}$ and $\mathbb{G}_{in}$ (left to right).

To provide an example of parallel composition, let us consider the graphs with interfaces $\mathbb{G} = \{a, p\} \to G \leftarrow \{n, m\}$ and $\mathbb{G}' = \{a, p\} \to G' \leftarrow \{m\}$ depicted in Figure 3.6. As said above, the graph $\mathbb{G}$ represents the graphical encoding of the processes $P = n[in\ m.\mathbf{0}]$, instead, as we will see later, the graph $\mathbb{G}'$ is the graph encoding of the processes $Q = m[out\ m.\mathbf{0}]$.

For the moment, the reader can ignore how these encodings are obtained. We only observe that in the graph $\mathbb{G}$ there is an edge *amb* representing the ambient n and an edge *in* simulating the capability $in\ m$. Analogously, in the graph $\mathbb{G}'$ there is an edge *amb* representing the ambient m and an edge *out* simulating the capability $out\ m$. Moreover, ambient names are represented by nodes of type $\circ$ that are in the output interfaces, and processes (subprocesses) are represented by graphs (subgraphs) that have as roots a pair of nodes $\langle \bullet, \diamond \rangle$. Only the root nodes $\langle \bullet, \diamond \rangle$ of the graphs representing the processes P and Q are in the input interfaces of the corresponding graphs. Moreover, as we can note, each subprocess is represented by a subgraph that has a different $\bullet$ root node, while sometimes subgraphs representing different sub-terms share the $\diamond$ root node. We will see later why this occurs.

The two graphs $\mathbb{G}$ and $\mathbb{G}'$ are compatible. Indeed, the type of the nodes belonging to both input interfaces coincides, and the same also holds for those nodes belonging to both output interfaces. Therefore, it is possible to compute the parallel composition of the graphs $\mathbb{G}$ and $\mathbb{G}'$, resulting in the graph with interfaces shown in Figure 3.7. It is easy to note that it is obtained by the union of the input interfaces and of the output interfaces, respectively, and the disjoint union of G and G', gluing the root nodes of both graphs and the nodes representing the name m. As we will see later, the graph with interfaces obtained by the parallel composition of $\mathbb{G}$ and $\mathbb{G}'$ represents the process obtained by making the parallel composition between P and Q, that is, the process $R = P \mid Q$.

3.3 Graph Rewriting

This section introduces the basic definitions for the DPO approach to the rewriting of (typed hyper-)graphs [21, 24] and graphs with interfaces. Some of them have already been presented in Section 2.3 in the more general setting of adhesive categories. However, since later on we are going to need the track function, we introduce a different definition of derivation between (systems as) graphs with interfaces using it.

Definition 3.11 (Graph production)**.** *A T-typed graph production $p : (L \xleftarrow{l} I \xrightarrow{r} R)$ is a production name p and a span of graph morphisms $(L \xleftarrow{l} I \xrightarrow{r} R)$ with l mono in T-$\mathbf{Graph}$. A T-typed graph transformation system (*GTS*) $\mathcal{G}$ is a pair $\langle T, P \rangle$, where T is a type graph and P is a set of productions with different names.*

Definition 3.12 (Graph derivation)**.** *Let $p : (L \xleftarrow{l} I \xrightarrow{r} R)$ be a T-typed graph production and G a T-typed graph. A match of p in G is a morphism $m_L : L \to G$. A direct derivation from G to H via production p and match m_L is a diagram as depicted in Figure 3.8, where (1) and (2) are pushouts in T-$\mathbf{Graph}$. We denote this derivation by $p/m : G \Longrightarrow H$, for $m = \langle m_L, m_I, m_R \rangle$, or simply by $G \Longrightarrow H$.*

Before giving the definition of derivation between graphs with interfaces, we introduce the notion of track function.

Definition 3.13 (Track function)**.** *Let p be a graph production and let $p/m : G \Longrightarrow H$ be a direct derivation, as in Figure 3.8. The track function $tr(p/m)$ associated with the derivation is the partial graph morphism $r^* \circ (l^*)^{-1} : G \to H$.*

The track function identifies the items before and after a derivation. It is used to give the definition of derivation between graphs with interfaces.

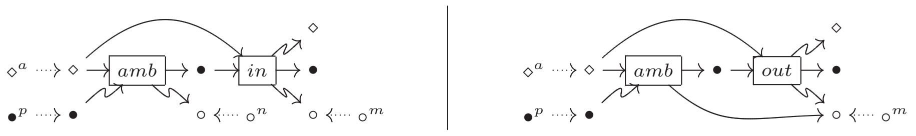


Figure 3.6: Graphs with interfaces $\mathbb{G}$ and $\mathbb{G}'$ (left to right).

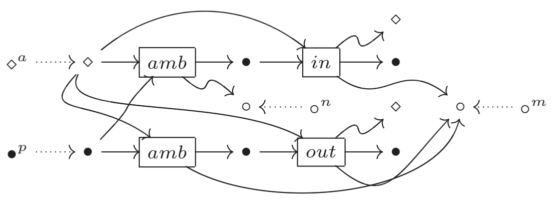


Figure 3.7: Graph with interfaces $\mathbb{G} \otimes \mathbb{G}'$.

Definition 3.14 (Graph with interfaces derivation)**.** *Let* $\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$ *and* $\mathbb{H} = J \xrightarrow{j'} H \xleftarrow{k'} K$ *be graphs with interfaces, and let* $p/m : G \Longrightarrow H$ *be a direct derivation such that the track function* $tr(p/m)$ *is total on* $j(J)$ *and* $k(K)$*. We say that* $p/m : \mathbb{G} \Longrightarrow \mathbb{H}$ *is a direct derivation of graphs with interfaces if* $j' = tr(p/m) \circ j$ *and* $k' = tr(p/m) \circ k$*.*

Intuitively, a derivation between graphs with interfaces is a direct derivation between the underlying graphs, such that inputs and outputs are preserved.

3.3.1 Parallel Independence and Confluence

We recall the classical notion of parallel independence, and states its connection with local confluence. A more general version (with sequential independence replacing confluence) can be found in [39, Section 3.3].

Definition 3.15 (Parallel independence)**.** *Let* $p_1/m_1 : G \Longrightarrow H_1$ *and* $p_2/m_2 : G \Longrightarrow H_2$ *be two direct derivations as in Figure 3.9. These derivations are* parallel independent *if there exists an* independence pair *among them, i.e., two graph morphisms* $i_1 : L_1 \rightarrow C_2$ *and* $i_2 : L_2 \rightarrow C_1$ *such that* $l_2^* \circ i_1 = m_{L_2}$ *and* $l_1^* \circ i_2 = m_{L_1}$*.*

Intuitively, two derivations as in Figure 3.9 are parallel independent if they act on disjoint items of the graph G, or at least on items that are simply read, and thus not deleted, by any of the two rule applications. The proposition below is a classical result relating parallel independence with rule sequentialisation (see e.g. [21]).

Proposition 3.2 (Confluence from independence)**.** Let $p_1/m_1 : G \Longrightarrow H_1$ and $p_2/m_2 : G \Longrightarrow H_2$ be two direct derivations as in Figure 3.9 such that they are parallel independent with independence pair $i_1 : L_1 \rightarrow C_2$ and $i_2 : L_2 \rightarrow C_1$. Then, there exists a graph H and two derivations $p_2/m_2^* : H_1 \Longrightarrow H$, with match $r_2^* \circ i_2$, and $p_1/m_1^* : H_2 \Longrightarrow H$, with match $r_1^* \circ i_1$, such that $tr(p_2/m_2^*) \circ tr(p_1/m_1) = tr(p_1/m_1^*) \circ tr(p_2/m_2)$.

Local confluence is thus implied by the standard notion of parallel independence. The notion is stronger than the corresponding property in e.g. term rewriting, since the preservation of the track function implies not only that the two derivations reach the same graph, but that the items of the starting graph are preserved. In particular, this implies that also the interface morphisms are preserved.

3.4 Graphical Encoding for Processes of Mobile Ambients

This section introduces a graphical encoding for processes of the mobile ambients calculus. First of all, we present a suitable type graph, depicted in Figure 3.10, and then we define an inductive encoding by exploiting the composition operators introduced in Definitions 3.7 and 3.9. This corresponds to a variant of

$$\begin{tikzcd}
p: & L \arrow[d, "m_L"'] \arrow[dr, phantom, "(1)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "m_I"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "m_R"] \\
 & G & C \arrow[l, "l^*"] \arrow[r, "r^*"'] & H
\end{tikzcd}$$

Figure 3.8: A direct derivation.

$$\begin{tikzcd}
R_1 \arrow[d, "m_{R_1}"'] & I_1 \arrow[l, "r_1"'] \arrow[r, "l_1"] \arrow[d, "m_{I_1}"] & L_1 \arrow[rr, dotted, no head] \arrow[drrrr, dotted] \arrow[dr, "m_{L_1}"] & & L_2 \arrow[dl, "m_{L_2}"'] \arrow[dllll, dotted] & I_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d, "m_{I_2}"] & R_2 \arrow[d, "m_{R_2}"] \\
H_1 & C_1 \arrow[l, "r_1^*"] \arrow[rr, "l_1^*"'] & & G & & C_2 \arrow[ll, "l_2^*"] \arrow[r, "r_2^*"'] & H_2
\end{tikzcd}$$

Figure 3.9: Parallel independence for $p_1/m_{L_1} : G \Longrightarrow H_1$ and $p_2/m_{L_2} : G \Longrightarrow H_2$.

the usual construction of the tree for a term of an algebra: names are interpreted as variables, so they are mapped to leaves of the graph and can be safely shared.

As we can see, in the type graph there are three types of node: the type of a node is denoted by its shape. Intuitively, a node of type $\circ$ represents an ambient name, while a graph that has as roots a pair of nodes $\langle \diamond, \bullet \rangle$ represents a process. More precisely, the node of type $\diamond$ represents the activating point for reductions of the process represented by the graph. We need two different types of node to model processes by graphs because each graph has to model both the syntactical and the activation dependences between the operators of a process. Indeed, in mobile ambients the nesting of operators does not reflect the activation dependences between them, since reductions can occur inside ambients. So, in order to model a process, we use $\bullet$ nodes to model the syntactical dependences between the operators of the process, and $\diamond$ nodes to model their activation ones.

Each edge of the type graph, except the go edge, simulates an operator of mobile ambients. Note that the act edge actually represents three edges, namely in, out and $open$. These three edges simulate the capabilities of the calculus, the amb edge represents the ambient operator, and the ν edge models the restriction operator [1]. Notice that there is no edge representing parallel composition. Finally, the go edge is a syntactical device for detecting the "entry" point for the computation. We need it later to simulate the reduction semantics of mobile ambients. It allows us to forbid the occurrence of a reduction underneath a capability operator.

All edges, except the go and ν edges, have the same type of source, that is the node list $\langle \diamond, \bullet \rangle$, while they have different types of target. In particular, the amb edge has the node list $\langle \bullet, \circ \rangle$ as target, while the in, out and $open$ edges have the same type of target, i.e. the node list $\langle \diamond, \bullet, \circ \rangle$. Note that these three latter edges have a node $\diamond$ in the target. This node represents the activating point for the reductions of the continuation of the capability. It is different from the activating point of the outermost capability operator, because the reductions of the continuation can occur only after the action regulated by the capability is executed. The amb edge instead has no node of type $\diamond$ in its target. In fact, the activating point for the reductions of the process inside an ambient is the same one of the outermost ambient. This occurs because process reductions permeate ambients. Unlike the other graphical operators, the ν operator has as root only one $\diamond$ node and does not have a $\bullet$ node. This modelling of the restriction operator comes from the fact that we consider this operator just as a scope operator. This solution, unlike that one proposed in [34], allows us to define an encoding of mobile ambients that captures the standard structural congruence of the calculus, dropping the *Cong-Zero-Res* axiom only.

Now we define a class of graphs such that all processes can be encoded into an expression containing

[1] Note that in the next chapter we are going to introduce a slightly different encoding. It will not use a graphical counterpart for the restriction operator, and thus it will be easier to use for our purpose of distilling a labelled transitions systems.

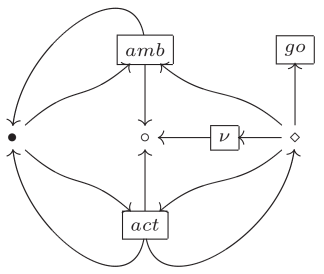


Figure 3.10: The type graph (for *act* $\in \{in, out, open\}$).

only those graphs as constants, and parallel and sequential composition as binary operators. Figures 3.11 and 3.12 depict these constant graphs. In particular, Figure 3.11 presents the graphs that correspond to the edges of the type graph. Figure 3.12 presents additional constant graphs needed for the formal presentation of our encoding. Note that in the graphs of the two figures we denote the input interface on the left and the output interface on the right. For example, the graph amb_n in the middle of Figure 3.11 has as input interface $\{a, p\}$ and as output interface $\{a, p, n\}$. Since a and p are constants used by our encoding, we assume that $p, a \notin \mathcal{N}$, while $n \in \mathcal{N}$ (where $\mathcal{N}$ is the set of names of mobile ambients).

In the following, we use $\mathbf{0}_{a,p}$ as shorthand for $\mathbf{0}_a \otimes \mathbf{0}_p$. Moreover, for a set of names Γ, we use id_Γ and $free_\Gamma$ as shorthands for $\bigotimes_{n\in\Gamma} id_n$ and $\bigotimes_{n\in\Gamma} free_n$, respectively. Note that both expressions are well defined, because the $\otimes$ operator is associative. The definition below introduces the encoding of processes into graphs with interfaces. It maps each finite process into a graph expression.

Definition 3.16 (Encoding for processes)**.** *Let P be a finite process and let Γ be a set of names such that $fn(P) \subseteq \Gamma$. The encoding of P, denoted by $[\![P]\!]_\Gamma$, is defined by structural induction according to the rules in Figure 3.13.*

Note that the encoding $[\![M.P]\!]_\Gamma$ represents the encoding of $in\ n.P$, $out\ n.P$ and $open\ n.P$, while act_n represents the in_n, out_n and $open_n$ graphs, respectively.

Our encoding addresses the α-conversion of restricted names by denoting them with $\circ$ nodes that are not in the image of the output morphism. The mapping is well-defined in the sense that the result is independent of the choice of the name m in the rule for restriction.

Moreover notice that in order to capture the axioms *Cong-Res-Par* and *Cong-Res-Amb*, our encoding extends the scope of each restriction operator to all the processes in parallel and to its parent ambient, respectively. Also, notice that the $\diamond$ root is the only root node that a graph representing a sub-process shares both with the graphs representing the other processes in parallel and with the graph representing its parent ambient. Therefore, the graphical operator modelling the restriction is linked only to the $\diamond$ root of the graph representing the process where it occurs. Note that linking the graphical operator ν also to the $\bullet$ root node would still allow to capture the structural axiom *Cong-Res-Par*, yet it would fail to recover the axiom *Cong-Res-Amb*. This means that the two congruent processes $(\nu n)m[P]$ and $m[(\nu n)P]$, for $n \neq m$, would be represented by different graphical encodings. This comes from the fact that in our encoding we do not use an edge to explicitly simulate the parallel operator $|$. Different processes in parallel are simply represented by the fact that they share the same root nodes $\langle \diamond, \bullet \rangle$. Instead, we use an explicit graphical operator to simulate the ambient operator, which shares with the process inside it only the $\diamond$ root node.

The encoding $[\![P]\!]_\Gamma$, where Γ is a set of names such that $fn(P) \subseteq \Gamma$, is a graph with interfaces $(\{a, p\}, \Gamma)$. We note that the mapping is not surjective. In fact, there are graphs with interfaces $(\{a, p\}, \Gamma)$ that are not in the image of the encoding. The encoding of a process P is the graph $[\![P]\!]_{fn(P)}$.

Example 3.1. Let us consider the example below, originally proposed in [16], which illustrates a form of planned dissolution of an ambient n

$$R = n[acid[out\ n.open\ n.P] \mid Q] \mid open\ acid.\mathbf{0}\ \ .$$

Figures 3.14 depicts the graph encoding $[\![R]\!]_{fn(R)}$. We represent the graph encodings for the processes P and Q by $\mathbb{G}_P$ and $\mathbb{G}_Q$, respectively. Moreover, for the sake of simplicity, we assume that the ambient names n and $acid$ do not belong to the set of free names of P and Q.

The leftmost edges, labelled *amb* and *open*, have the same roots, into which the names a and p are mapped. Those two edges represent the topmost operators of the two parallel components of the process.

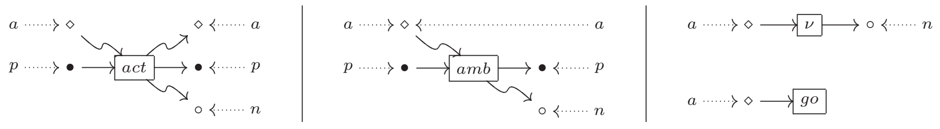


Figure 3.11: Graphs act_n (with $act \in \{in, out, open\}$); amb_n; ν_n and go (left to right).

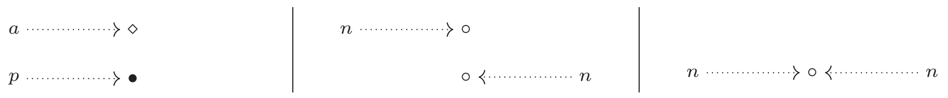


Figure 3.12: Graphs $\mathbf{0}_a$ and $\mathbf{0}_p$; $\mathbf{0}_n$ and $free_n$; and id_n (top to bottom and left to right).

The edges in the middle, representing from left to right the operators $acid[\_]$ and $out\ n.\_$, respectively, are linked to the same $\diamond$ root. Intuitively, this means that they have the same activating point of the outermost ambient, and hence the reductions can permeate the two ambients n and $acid$. Instead, the rightmost edge, labelled *open*, has a different $\diamond$ source that is the target of the edge *out*. Intuitively, this means that this capability *open* can be executed only after the action *out*.

The graphical encoding shown in the example above models a process where all the ambient names are free. The next example shows instead how our encoding models a process with restricted names. It also shows how our encoding is able to capture the structural axioms *Cong-Res-Res*, *Cong-Res-Par* and *Cong-Res-Amb*.

Example 3.2. Let S be the process $(\nu m)(\nu n)(m[n[P] \mid open\ n.Q] \mid open\ m.R)$, where $m \neq n$. The encoding $[\![S]\!]_{fn(S)}$ is depicted in Figure 3.15. We represent the graph encodings for the processes P, Q and R by $\mathbb{G}_P$, $\mathbb{G}_Q$ and $\mathbb{G}_R$, respectively. Moreover, for the sake of simplicity, we assume that the names m and n do not belong to the free names of P, Q and R.

The graph in Figure 3.15 encodes $(\nu n)(\nu m)(m[n[P] \mid open\ n.Q] \mid open\ m.R)$, as well as the process $(\nu m)((\nu n)(m[n[P] \mid open\ n.Q]) \mid open\ m.R)$ and furthermore also $(\nu m)(m[(\nu n)n[P] \mid open\ n.Q] \mid open\ m.R)$. The first two processes are congruent to S by the axioms *Cong-Res-Res* and *Cong-Res-Par*, respectively. The latter is congruent to the middle one thanks to the structural axiom *Cong-Res-Amb*.

The following theorem states that our encoding is sound and complete with respect to the structural congruence $\equiv$.

Theorem 3.1. *Let P, Q be processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. Then, $P \equiv Q$ if and only if $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$.*

The proof of Theorem 3.1 is shown in Section A.1 (Appendix A).

3.5 A Graph Transformation System for Mobile Ambients

This section presents a graph transformation system that models the reduction semantics of the mobile ambients calculus.

First of all, we enrich the encoding introduced in Definition 3.16 in order to avoid performing reductions underneath capability operators. To do this we attach a go edge to the $\diamond$ root node of each graph representing a process. The go edge is a syntactical device needed for detecting the "entry" point for the computation of the process. Given a process P and a set of names Γ such that $fn(P) \subseteq \Gamma$, its enriched encoding is the graph $[\![P]\!]_\Gamma \otimes go$. We denote it by $[\![P]\!]^{go}_\Gamma$.

Figure 3.16 presents the rules of the GTS $\mathcal{R}_{amb}$, which simulates the reduction semantics $\rightarrow$ introduced in Section 3.1. The GTS $\mathcal{R}_{amb}$ contains just three rules, namely p_{in}, p_{out} and p_{open}. They simulate the *Red-In*, *Red-Out* and *Red-Open* reductions, respectively. The action of the three rules is described by the

$$
\begin{array}{lcll}
[\![\mathbf{0}]\!]_\Gamma & = & \mathbf{0}_{a,p} \otimes free_\Gamma & \\
[\![n[P]]\!]_\Gamma & = & amb_n \circ (id_n \otimes [\![P]\!]_\Gamma) & \\
[\![M.P]\!]_\Gamma & = & act_n \circ (id_n \otimes [\![P]\!]_\Gamma) & \\
[\![(\nu n)P]\!]_\Gamma & = & (\nu_m \otimes [\![P\{m/n\}]\!]_{\Gamma\cup\{m\}}) \circ (\mathbf{0}_m \otimes id_\Gamma) & \text{for } m \notin \Gamma \\
[\![P \mid Q]\!]_\Gamma & = & [\![P]\!]_\Gamma \otimes [\![Q]\!]_\Gamma &
\end{array}
$$

Figure 3.13: Encoding for processes.

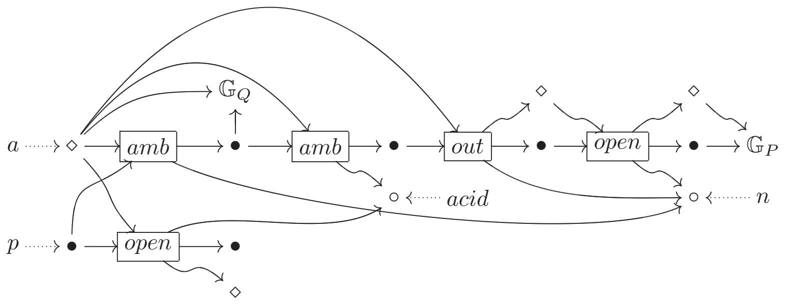


Figure 3.14: Graph encoding for the process $n[acid[out\ n.open\ n.P] \mid Q] \mid open\ acid.\mathbf{0}$.

node identifiers. These identifiers are of course arbitrary. They correspond to the actual elements of the set of nodes and are used to characterise the track function.

Now we discuss the rules of the GTS $\mathcal{R}_{amb}$. In order to give a clear explanation of the rule actions, we denote by amb_n an amb edge having in its target a $\circ$ node identified by n. Let us consider the p_{in} production. The p_{in} rule preserves the amb_m edge, removes the amb_n edge and re-creates this last one under amb_m. Note that, after the reduction, the in edge disappears and the nodes identified by 2_p and 3_p and by 1_a and 3_a are pair-wise coalesced. The former coalescing guarantees the "structural" integrity of the resulting graph, i.e., that all continuation processes are put in parallel; the latter ensures, as a side effect, that the $\diamond$ node 3_a under the in prefix is activated.

The p_{out} rule preserves the amb_m edge and removes the amb_n edge, too. It also re-creates this last one with the same source nodes of amb_m. Analogously to p_{in}, after the reduction the out edge disappears and the nodes identified by 3_p and 4_p and by 1_a and 4_a are pair-wise coalesced.

Finally, the p_{open} production removes both amb and $open$ edges. After the reduction, all the $\diamond$ nodes and all the $\bullet$ nodes are coalesced.

It seems noteworthy that three rules suffice for recasting the reduction semantics of mobile ambients. That is possible for two reasons. First, the closure of reduction with respect to contexts is obtained by the fact that graph morphisms allow the embedding of a graph within a larger one. Second, no distinct instance of the rules is needed, since graph isomorphism takes care of the closure with respect to structural congruence, and interfaces of the renaming of free names.

We now introduce the main theorems of the chapter. They state that our encoding is sound and complete with respect to the reduction relation $\rightarrow$.

Theorem 3.2 (Soundness)**.** *Let P, Q be processes and Γ a set of names, with $fn(P) \subseteq \Gamma$. If $P \rightarrow Q$, then $\mathcal{R}_{amb}$ entails a direct derivation $[\![P]\!]^{go}_{\Gamma} \Longrightarrow [\![Q]\!]^{go}_{\Gamma}$.*

Intuitively, a process reduction is simulated by applying a rule on an enabled event, that is, by a match covering a subgraph with the go operator on top.

Theorem 3.3 (Completeness)**.** *Let P be a process and Γ a set of names, with $fn(P) \subseteq \Gamma$. If $\mathcal{R}_{amb}$ entails a direct derivation $[\![P]\!]^{go}_{\Gamma} \Longrightarrow \mathbb{G}$, then there exists a process Q, such that $P \rightarrow Q$ and $\mathbb{G} = [\![Q]\!]^{go}_{\Gamma}$.*

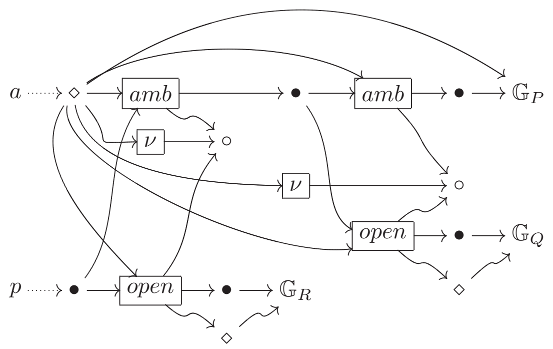
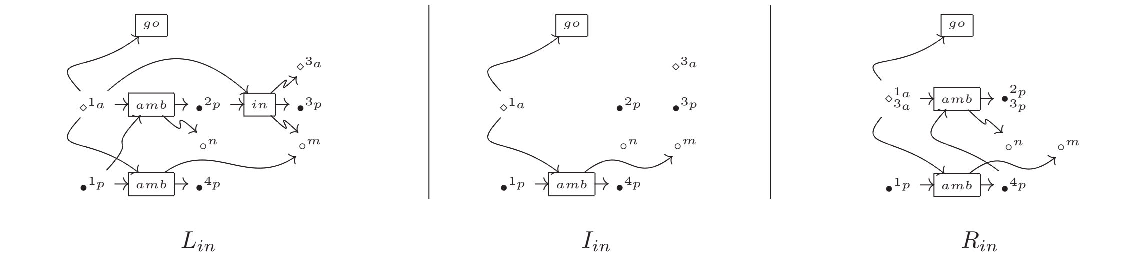
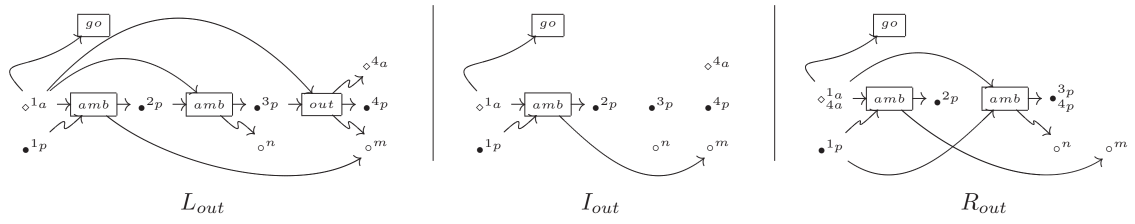


Figure 3.15: Graph encoding for the process $(\nu m)(\nu n)(m[n[P] \mid open\ n.Q] \mid open\ m.R)$.

$$n[in\,m.P|Q]|m[R] \to m[n[P|Q]|R]$$



$$m[n[out\,m.P|Q]|R] \to n[P|Q]|m[R]$$



$$open\,n.P|n[Q] \to P|Q$$

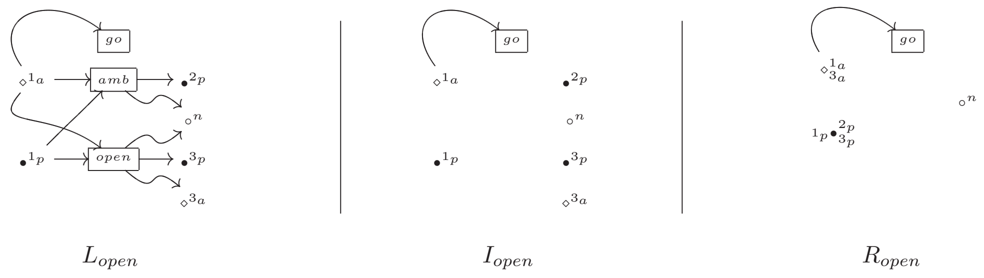


Figure 3.16: The rewriting rules p_{in}, p_{out} and p_{open} (top to bottom).

The proofs of Theorems 3.2 and 3.3 are shown in Section A.2 (Appendix A).

The correspondence holds since a rule is applied only if there is a match that covers a subgraph with the go operator on the top. This allows the occurrence of reductions inside activated ambients, but not inside capabilities. In fact, if an amb operator is activated, that is, its $\diamond$ source node has an outgoing go edge, then all operators inside it are activated too, because they have the same source node $\diamond$ as the amb operator. Differently, a reduction can not occur inside the outermost capability, because the activating point for the reductions of the continuation of a capability is different from the activating point of the outermost capability.

The following example shows the application of some rules of the GTS $\mathcal{R}_{amb}$ to the graph encoding for the process considered in Example 3.1.

Example 3.3. Let us consider again the process shown in Example 3.1

$$R = n[acid[out\ n.open\ n.P] \mid Q] \mid open\ acid.\mathbf{0}\ \ .$$

The graphical encoding for the process above is depicted in Figure 3.14. Its enriched encoding is instead presented in Figure 3.17, where the nodes are labelled in order to denote the track function of the derivation. The edge labelled *go* denote the entry point for the computation of the process.

Note that the two edges *amb*, the edge *out* and the outermost edge *open* can be involved in a reduction step because they have the same activation node with an outgoing *go* edge. Instead, the rightmost edge, labelled *open*, is not activated, since its $\diamond$ source is the target of another edge.

The application of the p_{out} rule to the graph in Figure 3.17 results in the graph in Figure 3.18, which is the actual encoding for the process $S = acid[open\ n.P] \mid n[Q] \mid open\ acid.\mathbf{0}$. In fact, this rewriting step simulates the transition $R \rightarrow S$.

Now, we can apply the p_{open} rule to the graph in Figure 3.18, and we obtain the graph in Figure 3.19. Note that this rewriting step simulates the transition $acid[open\ n.P] \mid n[Q] \mid open\ acid.\mathbf{0} \rightarrow open\ n.P \mid n[Q]$.

Finally, by applying the p_{open} rule to the graph in Figure 3.19, we get the graph in Figure 3.20. The derivation mimics the reduction $open\ n.P \mid n[Q] \rightarrow P \mid Q$.

The rewriting steps shown in the example above simulate a sequence of process reductions all occurring on the top. The next example shows how our encoding is able to simulate process reductions that are nested inside ambients.

Example 3.4. Let us consider the process previously shown in Example 3.2, namely, $S = (\nu m)(\nu n)(m[n[P] \mid open\ n.Q] \mid open\ m.R)$, where $m \neq n$ and m and n do not belong to the free names of P, Q and R. The encoding $[\![S]\!]_{fn(S)}$ is depicted in Figure 3.15, while the enriched encoding $[\![S]\!]^{go}_{fn(S)}$ is presented in Figure 3.21.

Two different applications of the p_{open} rule to the graph $[\![S]\!]^{go}_{fn(S)}$ are possible. The first application results in the graph on the left of Figure 3.22 and it simulates the process reduction nested inside the ambient m, namely, $S \rightarrow (\nu m)(\nu n)(m[P \mid Q] \mid open\ m.R)$. The other possible application of the p_{open} rule instead results in the graph on the right of Figure 3.22. This last rewriting step mimics the transition

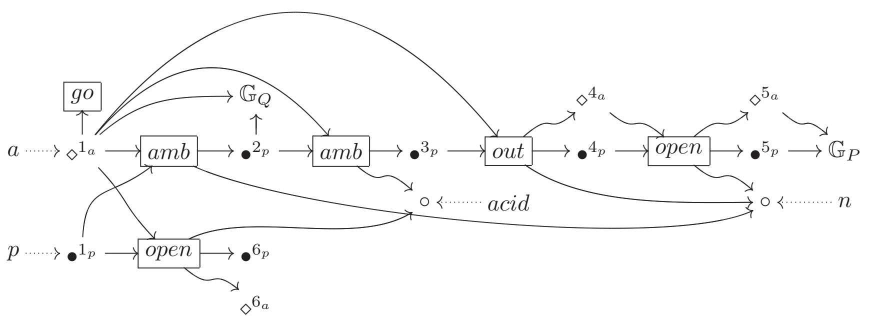


Figure 3.17: Graph encoding $[\![n[acid[out\ n.open\ n.P] \mid Q] \mid open\ acid.\mathbf{0}]\!]^{go}_{fn(R)}$.

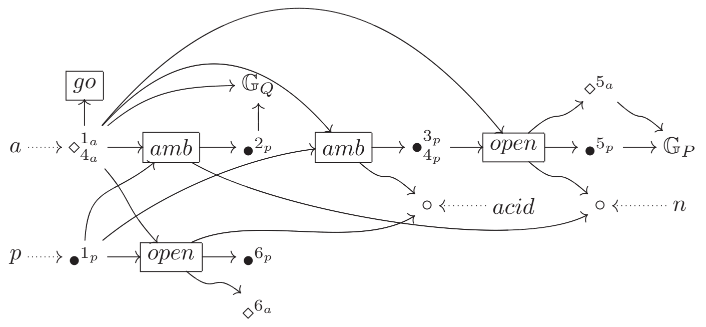


Figure 3.18: Graph encoding $[\![acid[open\ n.P] \mid n[Q] \mid open\ acid.\mathbf{0}]\!]^{go}_{fn(R)}$.

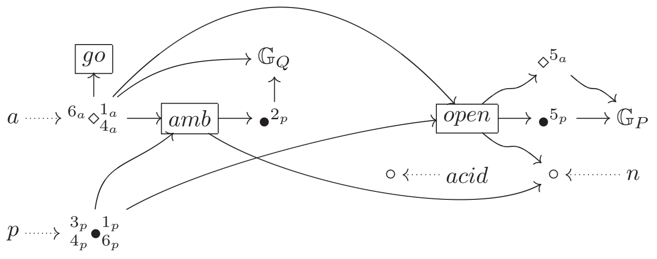


Figure 3.19: Graph encoding $[\![open\ n.P \mid n[Q]]\!]^{go}_{fn(R)}$.

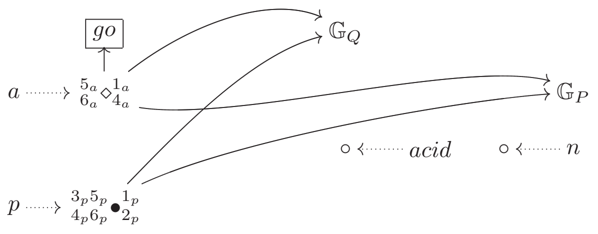


Figure 3.20: Graph encoding $[\![P \mid Q]\!]^{go}_{fn(R)}$.

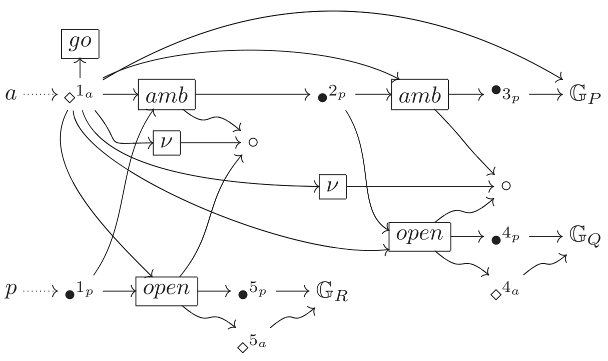


Figure 3.21: Graph encoding $[\![(\nu m)(\nu n)(m[n[P] \mid open\ n.Q] \mid open\ m.R)]\!]^{go}_{fn(S)}$.

$S \rightarrow (\nu m)(\nu n)(n[P] \mid open\ n.Q] \mid R)$. Now, it is possible to apply again the p_{open} rule to both graphs in Figure 3.22. The rewriting step obtained by applying the p_{open} rule to the graph on the left mimics the transition $(\nu m)(\nu n)(m[P \mid Q] \mid open\ m.R) \rightarrow (\nu m)(\nu n)(P \mid Q \mid R)$, while the rewriting step obtained by applying the p_{open} rule to the graph on the right simulates the transition $(\nu m)(\nu n)(n[P] \mid open\ n.Q] \mid R) \rightarrow (\nu m)(\nu n)(P \mid Q \mid R)$. Both the rewriting steps result in the graph in Figure 3.23.

3.6 Collecting useless restrictions

In Section 3.4 we introduced a graphical encoding for mobile ambients processes, proving its soundness and completeness. The price to pay was the dropping the axiom equating processes $(\nu x)\mathbf{0}$ and $\mathbf{0}$, since the encoding of the former has the occurrence of an edge which is missing in the one of the latter. This section shows how to recast the structural congruence $\equiv'$ of mobile ambients in terms of graph isomorphism. To this end, we introduce the GTS $\mathcal{R}^{\nu}$: it contains just the rewriting rule p_{ν} shown in Figure 3.24. Here the span of the graph morphisms is not presented explicitly, since it is obvious. The rule removes the useless occurrences of the restriction operator, i.e., such that the name it binds does not occur in the process. Indeed, in the graphical encoding this means that the node $\circ$ representing the restricted name is not shared with other operators. The rewriting rule removes only these $\circ$ nodes: it cannot be applied unless the node representing the name is isolated.

We start with a very simple technical lemma.

Lemma 3.1. *Let $p_{\nu}/m_1 : G \Longrightarrow H_1$ and $p_{\nu}/m_2 : G \Longrightarrow H_2$ be two distinct direct derivations. Then, these derivations are parallel independent.*

This result guarantees that the definition below is well-given.

Definition 3.17 (Normal form)**.** *Let $\mathbb{G}$ be a graph with interfaces. We call* normal form *of $\mathbb{G}$, in symbols $nf(\mathbb{G})$, the graph with interfaces obtained by applying as many times as possible the rewriting rule of the* GTS *$\mathcal{R}^{\nu}_{amb}$ to $\mathbb{G}$.*

In other words, the graph with interfaces $nf(\mathbb{G})$ is the normal form of $\mathbb{G}$ if and only if it is impossible to apply the rule of the GTS $\mathcal{R}^{\nu}_{amb}$ to $\mathbb{G}$.

The proposition below states that the normal form of our graphical encoding is sound and complete with respect to the process equivalence $\equiv'$.

Proposition 3.3. Let P, Q be processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. Then, $P \equiv' Q$ if and only if $nf([\![P]\!]_{\Gamma}) = nf([\![Q]\!]_{\Gamma})$.

The proof of Proposition 3.3 is shown in Section A.3 (Appendix A).

Example 3.5. Let us consider the process $T = (\nu m)(\nu n)(m[P \mid Q] \mid open\ m.R)$, where the names m and n do not belong to the free name of P, Q and R. The graphical encoding $[\![T]\!]^{go}_{fn(T)}$ is shown on the left of Figure 3.22. It is indeed equal to the encoding $[\![T]\!]^{go}_{fn(S)}$, since $fn(T) = fn(S)$. The normal form

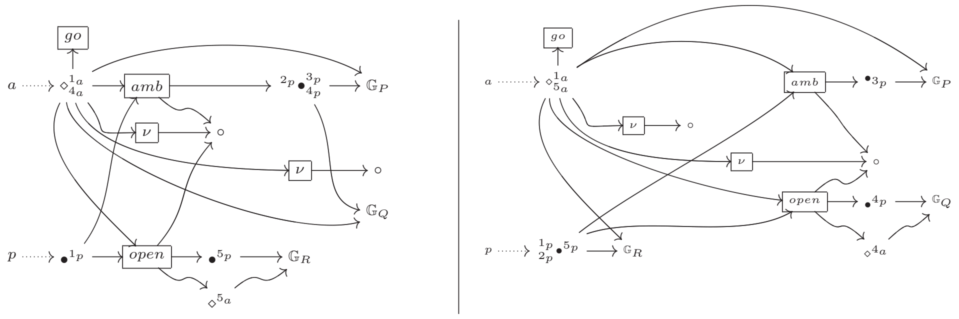


Figure 3.22: Graph encodings $[\![(\nu m)(\nu n)(m[P \mid Q] \mid open\ m.R)]\!]^{go}_{fn(S)}$ and $[\![(\nu m)(\nu n)([n[P] \mid open\ n.Q] \mid R)]\!]^{go}_{fn(S)}$ (left to right).

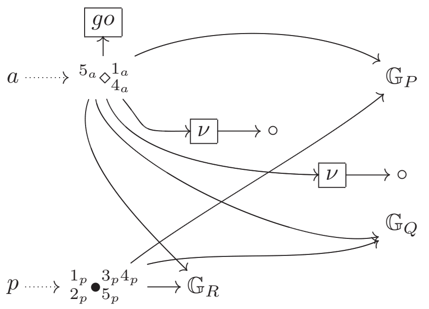


Figure 3.23: Graph encoding $[\![(\nu m)(\nu n)(P \mid Q \mid R)]\!]^{go}_{fn(S)}$.

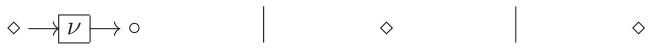


Figure 3.24: The rewriting rule p_ν for removing the useless restriction operators.

$nf([\![T]\!]^{go}_{fn(T)})$ is represented in Figure 3.25. It is obtained by applying only once the rewriting rule in Figure 3.24. Such a rule allows us to remove from the graph $[\![T]\!]_{fn(T)}$ the isolated node representing the useless restricted name n.

We now present the theorem stating that the normal form of our encoding is sound and complete with respect to the reduction relation $\rightarrow'$.

Theorem 3.4 (Soundness and Completeness)**.** *Let P, Q be processes and Γ a set of names, with $fn(P) \subseteq \Gamma$. If $P \rightarrow' Q$, then $\mathcal{R}_{amb}$ entails a direct derivation $nf([\![P]\!]^{go}_\Gamma) \Longrightarrow \mathbb{G}$ such that $nf(\mathbb{G}) = nf([\![Q]\!]^{go}_\Gamma)$.*

Let P be a process and Γ a set of names, with $fn(P) \subseteq \Gamma$. If $\mathcal{R}_{amb}$ entails a direct derivation $nf([\![P]\!]^{go}_\Gamma) \Longrightarrow \mathbb{G}$, then there exists a process Q, such that $P \rightarrow' Q$ and $nf(\mathbb{G}) = nf([\![Q]\!]^{go}_\Gamma)$.

The proof of Theorem 3.4 is shown in Section A.3 (Appendix A).

We close the section by presenting another simple technical lemma.

Lemma 3.2. *Let $p_\nu/m_1 : G \Longrightarrow H_1$ and $p/m_2 : G \Longrightarrow H_2$ be two (distinct) direct derivations, for any $p \in \mathcal{R}_{amb}$. Then, these derivations are parallel independent.*

Thus, the GTS $\mathcal{R}^\nu_{amb}$ given by the union of $\mathcal{R}^\nu$ and $\mathcal{R}_{amb}$ can be considered as a graphical implementation of the reduction semantics $\rightarrow'$, simultaneously allowing the normalisation of a process and the execution of a reduction step.

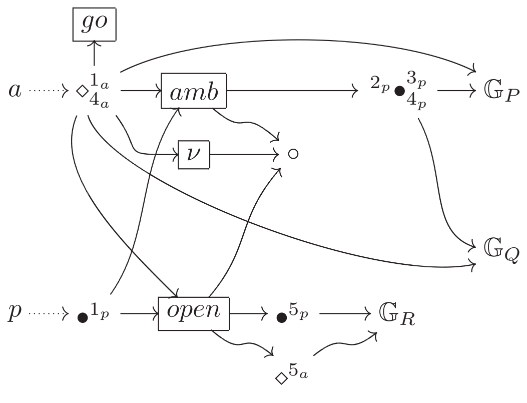


Figure 3.25: Normal form of the encoding $[\![(\nu m)(\nu n)(m[P \mid Q] \mid open\ m.R)]\!]^{go}_{fn(T)}$.

3.7 Concurrency and Interference

Our encoding may be exploited for defining a concurrent reduction semantics of the mobile ambients calculus. The role of the intermediate graph I in a rewriting rule is to characterise the elements of the graph to be rewritten that are read but not consumed by a direct derivation. Such a distinction is important when considering *concurrent* derivations, defined as equivalence classes of concrete derivations up to so-called *shift equivalence* [21], identifying (as for the analogous, better-known *permutation* equivalence of λ-calculus) those derivations that differ only for the scheduling of independent steps. Roughly, the equivalence states the interchangeability of two direct derivations $p_1/m_1 : G \Longrightarrow H$ and $p_2/m_2 : H \Longrightarrow M$ if they act either on disjoint parts of G, or on parts that are in the image of the intermediate graphs (in jargon, if they are *sequential independent* derivations).

As far as our encoding is concerned, the presence of the operator go linked to the $\diamond$ root node on the interface graph allows the simultaneous execution of several reductions. Indeed, the sharing of this operator allows the execution of several rewriting steps which act either on disjoint parts of the graph, or on parts that are in the image of the interface graphs. Note that the fact that the go operator is linked to the $\diamond$ root node also allows the simultaneous execution of different reductions which can occur both at top-level and inside ambients.

Let us consider the process $S = (\nu m)(\nu n)(m[n[P] \mid open\ n.Q] \mid open\ m.R)$, previously proposed in Examples 3.2 and 3.4. Its graphical encoding is presented in Figure 3.21. As shown in Example 3.4, two different rewriting steps starting from $[\![S]\!]_{fn(S)}$ are possible: the rewriting step simulating the opening of the restricted ambient m, and that one simulating the opening of the restricted ambient n. It is easy to notice that the two reductions are parallel independent. They indeed act only on disjoint items of the graph $[\![S]\!]_{fn(S)}$, or on items that are simply read, and thus not deleted, by any of the two rule applications, i.e., on items that are in the image of the graph I of the rule applied in both cases. Therefore, these two reductions can be executed simultaneously or, put differently, local confluence ensures that they give rise to two derivations (shown in Example 3.4) that differ only in the scheduling of the two steps. With respect to the solution proposed in [36], there is no need to apply any broadcasting rule to the graph graph $[\![S]\!]_{fn(S)}$. Those rules were needed there to communicate to the subprocesses the information about "being activated", and thus allowing the two reductions to be executed.

The definition of independence can be used to give a definition of interference. As explained in [46], an *interference* occurs when a derivation is corrupted by the execution of another derivation. Here authors identify two types of interferences which they call *plain interference* and *grave interference*. The former occurs when a process may execute the same interaction with two different partners, while the latter occurs when the two interactions are logically different. While the first type of interferences is sometimes desired, for example to model non-determinism, grave interferences can be considered "programming errors", as argued in [46].

In [46], both types of interference are defined informally. Authors use the notion of *redex* to denote the pair of ambients or processes involved in a reduction, therefore, an interference occurs when two or more redexes share one of the interactive partners. The problem is that different occurrences of the same subterms in a process are not identified and so, the mere notion of redex is not able to say which occurrence of the sub-term is used in presence of equal sub-terms in the process. For instance, let us consider the mobile ambient process $n[in\ m.P] \mid n[in\ m.P] \mid m[Q]$. In it we can identify two redexes, both formed by $n[in\ m.P] \mid m[Q]$, but it is obvious that we can not identify which occurrences of the subprocess $n[in\ m.P]$ they are actually using.

Giving a formal definition of interference is instead possible by using our graphical encoding. As shown previously, the reduction semantics of mobile ambients is modelled by a graph transformation system, and a process reduction is simulated by applying a rewriting rule, that is, by finding a match of the production in the graph representing the process. The notion of match exactly identifies the sub-terms involved in a reduction, therefore a formal definition of plain and grave interference can be introduced.

Definition 3.18 (Plain and grave interference)**.** *Let $p_1/m_1 : G \Longrightarrow H_1$ and $p_2/m_2 : G \Longrightarrow H_2$ be two direct derivations. We say that they* interfere *if they are not parallel independent. The interference is said* plain *if $p_1 = p_2$, and* grave *otherwise.*

Below we introduce some example of grave interference and we show how we can identify them by using the graphical encoding.

Example 3.6. Let us consider the process $R = open\ n.\mathbf{0} \mid n[in\ m.P] \mid m[Q]$, originally proposed in [46], where the names m and n do not belong to the free names of P and Q. Here the execution of the *open* reduction on the ambient n destroys the possibility to perform the *in* reduction on the ambient m, and vice versa. Indeed, both reductions act on the same ambient n by making changes on the structure of the process that destroy the possibility of performing the other reduction. Since the two interactions are logically different (in the first case we apply the *Red-Open* rule while in the second one we apply the *Red-In* rule), then a grave interference occurs. This is confirmed by the analysis of the graph in Figure 3.26 (representing the graphical encoding of the process R) and of the interface graphs of the applied rewriting rules. The graph in Figure 3.26 confirms the possibility of performing the p_{open} and p_{in} rules originating two derivations which simulate the two reductions above. It is also easy to note that matches of both rewriting rules share the ambient n. This ambient is not in the interface graphs of both rules, therefore it is consumed by them. This means that the two derivations are not parallel independent and hence a symmetric grave interference between them occurs.

Now let us consider the process $S = o[P] \mid n[in\ o.\mathbf{0} \mid m[out\ n.Q]]$, also proposed in [46], where the names o, n and m do not belong to the free names of P and Q. Here it is possible to execute the *in* reduction, allowing n to enter the sibling ambient o, or to execute the *out* reduction, allowing the ambient m to exit from n. Differently from the case above, the execution of a reduction does not destroy the possibility to perform the other one although it turns out to be corrupt, that is, its execution gives a different outcome from the one obtained by applying the same rule before the execution of the other reduction. This is confirmed by the analysis of the graph in Figure 3.27 (representing the graphical encoding of the process S) and of the interface graphs of the applied rewriting rules. It is easy to note that we can apply both the rules p_{in} and p_{out} originating two derivations which simulate the two reductions above. In this case the two matches share the ambient n, which is only read by the p_{out} rule but read and manipulated by the p_{in} rule. The two derivations are indeed not parallel independent and hence a grave interference occurs.

3.8 Related Work

The encoding presented before is not the only attempt proposed so far to give a graphical implementation of the mobile ambients calculus. The earliest proposals we are aware of are [30] and [36]. Our solution is reminiscent of the latter, lifting the use of unstructured graphs in the encoding of processes proposed there. Besides introducing a slender graph syntax (in accordance to [32]), the difference with the previous proposal lies in the chosen representation of the states: the lack of records for the activation points in [36] forced the introduction of suitable rules for forwarding the information about "being enabled" to subprocesses. The presence of such spurious rules, possibly inhibiting the execution of some reductions, made the correspondence between graph transformations and process reductions only weakly sound and complete (see e.g. [36, Theorems 5.3 and 5.4]). Thus, it made less meaningful the application of standard tools from graph transformation (such as the different parallelism theorems) for discussing about properties of process evolution. Therefore, also the use of the concurrent semantics of mobile ambients in the study of the behaviour of a process with respect to dynamic properties such as non-interference [46] was less appealing

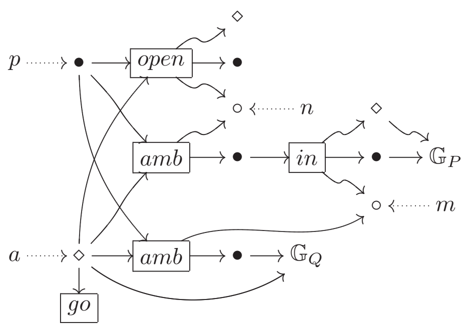


Figure 3.26: Graph encoding $[\![open\ n.\mathbf{0} \mid n[in\ m.P] \mid m[Q]]\!]^{go}_{fn(R)}$.

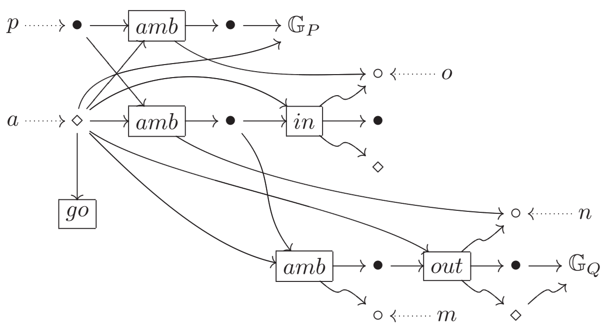


Figure 3.27: Graph encoding $[\![o[P] \mid n[in\ o.\mathbf{0} \mid m[out\ n.Q]]]\!]^{go}_{fn(S)}$.

and intuitive. Our chosen state representation allows instead for the reuse of such techniques, as surveyed in [32] for the π-calculus.

As far as other proposals for graphical implementation are concerned, we are aware of [18, 30], using the so-called Synchronised Hyper-edge Replacement (SHR) framework, as well as of [55], in the mold of the standard DPO approach. Moreover, in [41] authors outline an encoding of mobile ambients by bigraphs. They however leave to future work the detailed study of their solution, which they intend to exploit for the derivation of a labelled transition system for the calculus.

In general, those SHR solutions are eminently hierarchical, meaning that each edge/label is itself a structured entity, and possibly a graph. More precisely, "sequential processes become edge labels: when an action is performed, an edge labelled by $M.P$ is rewritten as the graph corresponding to P" [30, p. 11]. We believe that this is less adequate for calculi such as mobile ambients, where the topology of the systems plays a major role in discussing e.g. about distributed implementation and parallel execution of reductions [46], as witnessed by the results shown in Sections 3.7 of this chapter. Moreover, the expressive power of the SHR framework is achieved via a rather complex mechanism for rule application, less intuitive and simple than the basic DPO matching of our solution.

As far as [55] is concerned, the main difference with respect to our proposal is in the use of a process representation where the nesting of ambients is made explicit by the presence of suitable edges, instead of being implicit in the representation of each process, as in our proposal. The resulting encoding of processes is thus centralised, and this condition results in a complex set of graph transformation rules. Moreover, the encoding of process reduction is sound, yet not complete, thus not allowing the reuse of tools for system analysis that we mentioned earlier.

3.9 Graphical Encoding for Asynchronous CCS

In this section we present an encoding for the finite fragment of the asynchronous CCS (ACCS). Differently from [42, 51], where processes are encoded into bigraphs, here as for mobile ambients we use unstructured graphs. In particular, we adapt the encoding for the synchronous CCS presented in [7]), also modelling the reduction semantics of the calculus via a set of DPO rules. We introduce such an encoding because it allows us to derive the IPO LTS for the ACCS (as shown in Chapter 4), which we will use to establish the adequacy of the results we will present in next chapters.

Asynchronous CCS. Here we shortly introduce ACCS as a fragment of asynchronous π (with no name passing). We adopt the presentation in [1] that allows the non deterministic choice for input prefixes (a feature missing in [17, 14]). Moreover, to simplify the presentation, we avoid to consider infinite processes.

The syntax of ACCS is shown in Figure 3.28: $\mathcal{N}$ is a set of *names*, ranged over by $a, b, \ldots$, with $\tau \notin \mathcal{N}$. We let $P, Q, \ldots$ range over the set $\mathcal{P}$ of processes and $M, N, \ldots$ over the set $\mathcal{S}$ of summations. With respect to synchronous CCS, the calculus lacks output prefixes: process $\overline{a}$ is thought of as a message, available on a communication media named a, that disappears after its reception. The *free names* $fn(P)$ of a process P are defined as usual.

Processes are taken up to a *structural congruence* (Figure 3.28), denoted by $\equiv$. The *reduction relation* is the least relation $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$, closed under $\equiv$, inductively generated by the rules in Figure 3.28. The

$$P ::= \bar{a},\ P_1 \mid P_2,\ (\nu a)P,\ M \qquad M ::= \mathbf{0},\ \alpha.P,\ M_1 + M_2 \qquad \alpha ::= a,\ \tau$$

$$\begin{array}{lll}
P \mid Q \equiv Q \mid P & (P \mid Q) \mid R \equiv P \mid (Q \mid R) & P \mid \mathbf{0} \equiv P \\
M + N \equiv N + M & (M + N) + O \equiv M + (N + O) & M + \mathbf{0} \equiv M \\
(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P & (\nu a)(P \mid Q) \equiv P \mid (\nu a)Q \quad \text{if } a \notin fn(P) & (\nu a)\mathbf{0} \equiv \mathbf{0} \\
 & (\nu a)P \equiv (\nu b)(P\{^b/_a\}) \ \text{ if } b \notin fn(P) &
\end{array}$$

$$\bar{a} \mid (a.P + M) \to P \qquad \tau.P + M \to P \qquad \frac{P \to Q}{(\nu a)P \to (\nu a)Q} \qquad \frac{P \to Q}{P \mid R \to Q \mid R}$$

Figure 3.28: Syntax, structural congruence and reduction relation of ACCS.

$$a.P + M \xrightarrow{a} P \qquad \tau.P + M \xrightarrow{\tau} P \qquad \bar{a} \xrightarrow{\bar{a}} \mathbf{0}$$

$$\frac{P \xrightarrow{\mu} Q \quad a \notin n(\mu)}{(\nu a)P \xrightarrow{\mu} (\nu a)Q} \qquad \frac{P \xrightarrow{\mu} Q}{P \mid R \xrightarrow{\mu} Q \mid R} \qquad \frac{P \xrightarrow{a} P_1 \quad Q \xrightarrow{\bar{a}} Q_1}{P \mid Q \xrightarrow{\tau} P_1 \mid Q_1}$$

Figure 3.29: Labelled semantics of ACCS

interactive semantics for ACCS is instead given by the relation over processes up to $\equiv$, obtained by the rules in Figure 3.29. We let μ range over the set of labels $\{\tau, a, \bar{a} \mid a \in \mathcal{N}\}$: the names of μ, denoted by $n(\mu)$, are defined as usual. Differently from synchronous calculi, sending messages is non-blocking. Hence, an observer might send messages without knowing about their reception, and inputs are thus deemed as unobservable. This is mirrored in the notion of asynchronous bisimilarity [1].

Definition 3.19 (Asynchronous bisimulation)**.** *A symmetric relation $\mathcal{R}$ is an* asynchronous bisimulation *if whenever $P \,\mathcal{R}\, Q$ then*

- *if $P \xrightarrow{\tau} P'$ then $Q \xrightarrow{\tau} Q'$ and $P' \,\mathcal{R}\, Q'$,*
- *if $P \xrightarrow{\bar{a}} P'$ then $Q \xrightarrow{\bar{a}} Q'$ and $P' \,\mathcal{R}\, Q'$,*
- *if $P \xrightarrow{a} P'$ then either $Q \xrightarrow{a} Q'$ and $P' \,\mathcal{R}\, Q'$ or $Q \xrightarrow{\tau} Q'$ and $P' \,\mathcal{R}\, Q' \mid \bar{a}$.*

Asynchronous bisimilarity $\sim^A$ *is the largest asynchronous bisimulation.*

Graphical Encoding for ACCS. We do not present the formal definition of the graphical encoding of ACCS, since it is analogous to the one for the synchronous version of the calculus presented in [7, Definition 9]: it differs only for the choice of the typed graph T_A, depicted in Figure 3.30. We remark that choosing a graph typed over T_A means to consider graphs where each node (edge) is labelled by a node (edge) of T_A, and the incoming and outcoming tentacles are preserved.

Intuitively, a graph having as root a node of type $\bullet$ ($\diamond$) corresponds to a process (respectively a summation), while each node of type $\circ$ basically represents a name. Indeed, even if the encoding could be defined by means of the two operators on typed graphs with interfaces defined in Section 3.2.1, for ACCS the situation is summed up by saying that a typed graph with interfaces is the encoding of a process P if its underlying graph is almost the syntactic tree of P: each internal node of type $\bullet$ has exactly one incoming edge, except for the root, to which an edge labelled *go* is attached.

Going back to the type graph, the edge *rcv* (*snd*) simulates the input prefix (output operator, respectively), while there is no edge for the parallel composition, non-deterministic choice and restriction operators. Edge *c* is a syntactical device for "coercing" the occurrence of a summation inside a process context, while similarly to the mobile ambients encoding, the edge *go* detects the "entry" point of the computation.

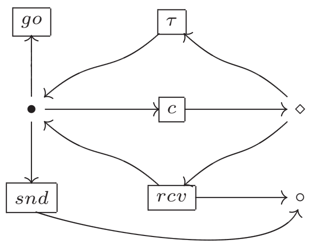


Figure 3.30: Type graph T_A.

In this case it avoids to perform any reduction below the outermost prefix operators: it is needed to properly simulate the reduction semantics of the calculus.

The encoding of a process P, with respect to a set of names Γ including the free names of P, is a graph with interfaces $(\{p\}, \Gamma)$. It is sound and complete with respect to the structural congruence of the calculus, that is, two processes are equivalent if and only if they are mapped into isomorphic graphs.

Figure 3.31 depicts the graph encoding for the process $P = (\nu b)(\bar{b} \mid b.\bar{a} + a)$. The two leftmost edges labelled c and snd have the same root, into which the node p of the interface is mapped. They are the top edges of the two subgraphs representing the parallel components of the process. In particular, the edge labelled snd represents the output over the restricted channel b, namely $\bar{b}$, while the c edge is the syntactical operator denoting that its subgraph represents a summation, that is, $b.\bar{a} + a$. The two leftmost edges of this last subgraph, both labelled rcv, model the two input prefixes b and a, while the rightmost snd edge represents the operator $\bar{a}$. Note that the channel name a is in the output interface since it is free in P, while the bound name b does not belong to the interface.

A Graph Transformation System for ACCS. Figure 3.32 shows the two rules of the GTS $\mathcal{R}_{ACCS}$, which simulates the reduction semantics of the calculus introduced above. It contains just two rules, namely p_{com} and p_τ, which mirror the two axioms of the reductions relation in Figure 3.28. Also in this case, as for mobile ambients, the action of the rules is described by the node identifiers. Note that a soundness and completeness result of our encoding with respect to reductions is easily obtained (see [7, Proposition 2]). Note that the correspondence must account for the discarding of sub-processes, due to the resolution of non-deterministic choices: after a DPO derivation there can be parts of the graph (representing the discarded components) that are not reachable from the root. Therefore, if a process P reduces to Q, we will not have that its graphical encoding will reduce to the encoding of Q with a DPO step, but it will reduce to a graph whose subgraph reachable from the root coincides with the encoding of Q. Vice versa, if the graphical encoding of a process P executes a DPO derivation and reaches the graph G, then there exist Q such that P reduces to Q and the subgraph of G reachable from the root coincides with the graphical encoding of Q.

3.10 Summary

We presented two graphical encodings for finite processes respectively of mobile ambients and asynchronous CCS. Each of them is sound and complete with respect to the operational semantics of the calculus it encodes: both are based on unstructured graphs and standard DPO approach tools, thus allowing for the reuse of analysis techniques from the graph transformation mold, along the lines of the graphical encodings presented in [32, 36].

The graphical encoding of the asynchronous CCS is basically an adaptation of the one for the synchronous version of the calculus [7]. As said in the introduction of this chapter, we presented the encoding,

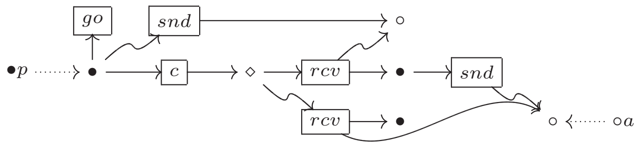


Figure 3.31: Encoding for the process $(\nu b)(\bar{b} \mid b.\bar{a} + a)$.

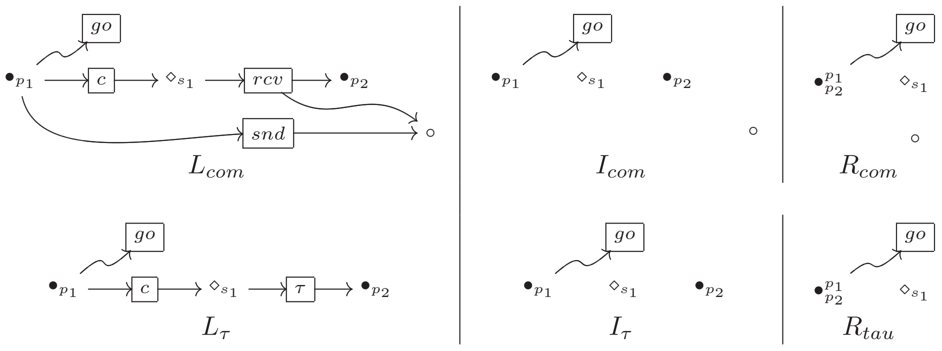


Figure 3.32: The productions $p_{com} : L_{com} \hookleftarrow I_{com} \rightarrow R_{com}$ and $p_\tau : L_\tau \hookleftarrow I_\tau \rightarrow R_\tau$.

since it allows us to derive the IPO LTS for the ACCS (as shown in Chapter 4), which is used to establish the adequacy of some results we will present in the next chapters.

As far as the encoding for mobile ambients, it has the ability to model the syntactic structure of a process and to keep track of its activation points, that is, of those ambients where reductions may actually take place. Therefore, it allows a simple and faithful modelling of the reduction semantics of mobile ambients. We considered the original presentation of the calculus, by discarding the communication primitives, as well as recursive expressions: both could be tackled along the lines of the solution in [32].

The article also offers a list of applications for the graphical encoding of π-calculus [32, Section 8], which could be immediately lifted to our encodings. They range from the use of graphs for verifying system properties expressed by spatial logic to the use of the *borrowed contexts* approach for deriving a labelled transition system for the encoded calculi.

Among them, we focused on the use of the borrowed contexts mechanism: a thorough study of the labelled transition system for mobile ambients obtained by exploiting the borrowed context technique is indeed presented in the next chapter. It should be remarked that this array of applications is possible also for mobile ambients thanks to our graphical implementation, where the tree structure of a process is decoupled from its activation points.

Chapter 4

RPO semantics for mobile ambients and asynchronous CCS

This chapter presents two case studies on the synthesis of LTSs for process calculi, choosing as testbed mobile ambients and asynchronous CCS, respectively introduced in Sections 3.1 and 3.9.

Both proposals are based on (a slight variant of) the graphical encodings of both calculi presented in the previous chapter, where each process is mapped into a graph equipped with suitable interfaces, such that the denotation is fully abstract with respect to the usual structural congruence. Graphs with interfaces are amenable to the borrowed contexts synthesis mechanism, which is an instance of G-relative pushouts. The mechanism allows the effective construction of a labelled transition systems that has graphs with interfaces as both states and labels, and such that the associated bisimilarity is automatically a congruence.

Here we concentrate on mobile ambients, by focusing on the analysis of the derived labelled transition system over (processes as) graphs with interfaces. In particular, we first use the labelled transition system on graphs to recover a suitable one directly defined over the structure of mobile ambients processes, and we then exploit it to define a set of inference rules (in the SOS style) capturing the same operational semantics for the calculus. The chapter is rounded up by a comparison with an alternative proposal by Rathke and Sobociński described in [60] (also inspired by the RPO technique).

Also as far as the asynchronous CCS is concerned, we introduce an IPO LTS for the calculus. We do not show all steps needed to obtain it, because the procedure we used is very similar to the one exploited in [7] for the synchronous version of the calculus. We present it just because it will be useful in some examples presented in the next chapters.

The chapter is organized as follows. Section 4.1 shortly introduces an extended syntax for the mobile ambients calculus, needed for the presentation of the operational semantics in Section 4.6. Section 4.2 recalls the DPO approach to rewriting on graphs with interfaces, as well as the associated BC technique for distilling an LTS. Then, in Section 4.3 we discuss a graphical encoding for the extended mobile ambients processes, which is the basis for a graph transformation system for mobile ambients that simulates process reduction, defined in Section 4.4. In turn, these are needed for the presentation in Section 4.5 of an LTS for graphs with interfaces representing mobile ambients processes, obtained by means of the BC synthesis mechanism. Furthermore, the LTS over graphs is exploited in Section 4.6 to introduce an LTS defined directly over processes of the mobile ambients calculus. In Section 4.7 we then present a novel description of the distilled LTS by means of a set of inference rules, given according to the SOS style. And finally, this SOS characterization is used in Section 4.8 to formally prove the correspondence between our proposal and Rathke and Sobociński's. Finally, before summarizing the chapter (Section 4.10), in Section 4.9 we briefly present a labelled transition system for the asynchronous CCS, synthesized by applying the BC technique to the graphical encoding introduced in Section 3.9.

4.1 Extended Mobile Ambients

This section shortly introduce an extended version of the mobile ambients calculus (previously introduced in Section 3.9): we need it for the presentation of the operational semantics in Section 4.6. There, indeed, we will present an LTS having as target states processes with underspecified subprocesses and/or ambient names which can be further instantiated. Therefore, we introduce an extended syntax that allows us to build processes containing *process variables* and *name variables*.

Definition 4.1 (Extended processes)**.** *Let $\mathcal{N}$ be a set of* names *ranged over by $m, n, u, \ldots$ and let $\mathcal{X} = \{X, Y, \ldots\}$ and $\mathcal{V} = \{x, y, \ldots\}$ be respectively a set of* process variables *and a set of* name variables*. An* extended process *is a term generated by the syntax in Figure 4.1.*

$$P ::= \mathbf{0}, n[P], M.P, (\nu n)P, P_1|P_2, X, x[P] \qquad\qquad M ::= in\ n, out\ n, open\ n$$

Figure 4.1: Extended syntax of mobile ambients.

Intuitively, an extended process such as $x[P]|X$ represents an underspecified process, where either the process X or the name of the ambient $x[-]$ can be further instantiated.

Definition 4.2 (Pure and well-formed extended processes)**.** *A* pure *process is an extended process such that no process or name variable occurs in it. A* well-formed *process is an extended process such that no process or name variable occurs more than once. We let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of pure processes; and $P_\epsilon, Q_\epsilon, R_\epsilon, \ldots$ over the set $\mathcal{P}_\epsilon$ of well-formed processes.*

We use the standard definitions for the set of free names of a pure process P, denoted by $fn(P)$, and for α-convertibility, with respect to the restriction operators (νn). As for the general definition, variables carry no name, hence $fn(x[P_\epsilon]) = fn(P_\epsilon)$ and $fn(X) = \emptyset$. Later on we are also going to need the set of name and process variables occurring in a process, defined as expected and denoted as $nv(P_\epsilon)$ and $pv(P_\epsilon)$.

Moreover, we consider a family of *substitutions*, which may replace a process/name variable with a pure process/name, respectively. Substitutions avoid name capture: for a pure process P, the expression $(\nu n)(\nu m)(m[X]|x[\mathbf{0}])\{^m/_x, ^{n[P]}/_X\}$ corresponds to the pure process $(\nu p)(\nu q)(q[n[P]]|m[\mathbf{0}])$, for names $p, q \notin \{m\} \cup fn(n[P])$.

The semantics of the calculus is given by the reduction relation and the structural congruence $\equiv$ defined on pure processes and respectively presented in Figures 3.3 and 3.2.

4.2 DPO Rewriting for Graphs with Interfaces

This section recalls the *double-pushout* (DPO) approach to the rewriting of graphs with interfaces. Note that the definition of DPO derivation for systems with interface has already been introduced in Section 2.3, by considering the more general setting of adhesive categories. Here we instantiate them for the case of graphs with interfaces in order to give the operational intuition of the production application, which will be useful later on.

In the following we use the notion of graph production introduced in Definition 3.11.

Definition 4.3 (Derivation of graphs with interfaces)**.** *Let $J \to G$ and $J \to H$ be two graphs with interfaces. Given a production $p : L \leftarrowtail I \longrightarrow R$, a* match *of p in G is a mono $m : L \rightarrowtail G$. A* direct derivation *from $J \to G$ to $J \to H$ via p and m is a diagram as depicted below, where (1) and (2) are pushouts and the bottom triangles commute. In this case we write $J \to G \Longrightarrow J \to H$.*

$$\begin{tikzcd}
L \arrow[d, tail, "m"'] & I \arrow[l, tail, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dl, phantom, "(1)"] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\
G & C \arrow[l, tail] \arrow[r] & H \\
& J \arrow[ul] \arrow[u, "k"] \arrow[ur] &
\end{tikzcd}$$

Operationally, applying a production p to a graph with interfaces $J \to G$ consists of three steps. First, the (injective) match $m : L \to G$ is chosen, providing an occurrence of L in G. Then, all the items of G matched by $L - l(I)$ are removed, leading to the *context graph* C. If C is well-defined, and the resulting square is indeed a pushout, the items of $R - r(I)$ are finally added to C, further coalescing those nodes and edges identified by r, obtaining the derived graph H.

The morphism $k : J \to C$ which makes the left triangle commute is unique, whenever it exists. If such a morphism does not exist, then the rewriting step is not feasible. Moreover, note that the standard DPO derivations (Definition 3.12) can be seen as a special instance of these, obtained considering as interface J the empty graph.

Note that here we require that the match m has to be mono. This condition is going to be necessary since it is needed for the application of BC rewriting. Note also that we do not report explicitly this notion, since it is the same of the one introduced in Section 2.3 for the more general setting of adhesive categories.

4.3 Graphical Encoding for Extended Mobile Ambients Processes

This section introduces the graphical encoding for the extended mobile ambients processes. It is very similar to the one presented in Section 3.4, where only processes over the standard syntax are considered. The only difference is that here we consider extended processes and, in order to apply the borrowed context technique, we need to have graphs with only one interface. Moreover, in order to simplify the encoding, we drop the graphical restriction operator. The lack of restriction operators is dealt with simply by manipulating the interfaces of graphs, that is, by denoting restricted names by name nodes that are not in the interface. As discussed later in this section, this new encoding forces us to consider a slightly different structural congruence containing the axioms in Figure 4.8, but on the other side it allows us to obtain a slender graphical encoding which is simpler to manipulate.

Figure 4.2 shows the type graph T_M that we consider: it differs from that one of Section 3.4 only for the absence of the restriction operator.

The intuitive meaning of nodes and edges is exactly the same: a node of type $\circ$ represents an ambient name, while a graph that has as roots a pair of nodes $\langle \diamond, \bullet \rangle$ represents a process, where $\diamond$ precisely denotes the activating point for reductions of the process. As far as edges are concerned, each of them, except the go edge, simulates an operator of mobile ambients, while the go edge is a syntactical device for detecting the "entry" point for the computation. We need it to simulate the reduction semantics of the calculus.

The well-formed processes are encoded into expressions, which as constants besides containing the graphs in Figures 3.11 and 3.12 (also used in the encoding in Section 3.4) also contain the graphs in Figure 4.3. Parallel and sequential composition (respectively Definitions 3.7 and 3.9) are instead the only binary operators which are used. We assume a family $\{a, p\} \uplus \{X_a, X_p \mid X \in \mathcal{X}\}$ with no intersection with $\mathcal{N}$.

In the following, besides using $\mathbf{0}_{a,p}$ and $id_{a,p}$ as shorthands for $\mathbf{0}_a \otimes \mathbf{0}_p$ and $id_a \otimes id_p$, respectively, we similarly exploit $\mathbf{0}_X$ and id_X which stand for $\mathbf{0}_{X_a} \otimes \mathbf{0}_{X_p}$ and $id_{X_a} \otimes id_{X_p}$. Moreover, for a set of names Γ, we use $\mathbf{0}_\Gamma$ and id_Γ as shorthands for $\bigotimes_{n \in \Gamma} \mathbf{0}_n$ and $\bigotimes_{n \in \Gamma} id_n$, respectively; and for a process P_ϵ we let $id_{pv(P_\epsilon)}$ stand for $\bigotimes_{X \in pv(P_\epsilon)} id_X$.

The definition below introduces the encoding of extended processes (with no occurrence of name variables) into graphs with interfaces, mapping a process into a graph expression. Note that the encoding $[\![M.P_\epsilon]\!]_\Gamma$ represents the encoding of $in\ n.P_\epsilon$, $out\ n.P_\epsilon$ and $open\ n.P_\epsilon$, while act_n represents the in_n, out_n

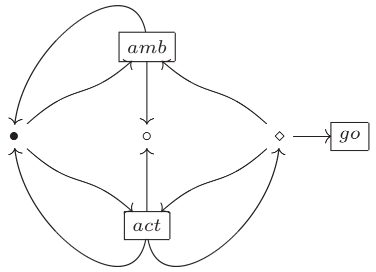


Figure 4.2: The type graph T_M (for *act* $\in \{in, out, open\}$).

and $open_n$ graphs, respectively.

Definition 4.4 (Encoding for processes)**.** *Let P_ϵ be a well-formed process with no occurrence of name variables and let Γ be a set of names such that $fn(P_\epsilon) \subseteq \Gamma$. The* encoding *of P_ϵ, denoted by $[\![P_\epsilon]\!]_\Gamma$, is defined by structural induction according to the rules in Figure 4.4.*

Given a well-formed process P and a set of names Γ, such that $fn(P_\epsilon) \subseteq \Gamma$, its encoding $[\![P_\epsilon]\!]_\Gamma$ is a graph with interfaces $(\{a,p\} \uplus \{X_a, X_p \mid X \in pv(P_\epsilon)\} \uplus \Gamma, \emptyset)$. Moreover, as in the encoding of Section 3.4, its *enriched encoding* is the graph $[\![P_\epsilon]\!]_\Gamma \otimes go$, which we denote by $[\![P_\epsilon]\!]^{go}_\Gamma$. Intuitively, it is obtained by attaching a go edge to the $\diamond$ root node of each graph representing a process.

Example 4.1. Consider the pure process $R = n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]$, previously introduced in Section 3.2.2. Figure 4.5 shows the graphical encoding $[\![R]\!]_{\{n,m\}}$ according to Definition 4.4. It is very similar to the one shown in Figure 3.7 and obtained by considering Definition 3.16: the only difference is that here names nodes representing free names are in the input interface together with root nodes $\bullet$ and $\diamond$.

Now, consider the pure process $S = (\nu n)R$. Its enriched encoding $[\![S]\!]^{go}_{\{m\}}$ is shown in Figure 4.6. This is obtained from $[\![R]\!]_{\{n,m\}}$ in two steps: at first, the node n is removed from the interface (obtaining $[\![S]\!]_{\{m\}}$); and then, the go edge is attached to the activation node $\diamond$ (finally obtaining $[\![S]\!]^{go}_{\{m\}}$).

Let us focus on the first step: by definition, $[\![S]\!]_{\{m\}} = (new_n \otimes id_m \otimes id_{a,p}) \circ [\![R]\!]_{\{n,m\}}$. The graph with interface $(new_n \otimes id_m \otimes id_{a,p})$ has the same underlying body of $id_{n,m,a,p}$, but the name n is missing from the input interface: the sequential composition of it with a graph having interface $\{n,m,a,p\}$ results into the same graph but without n among its inputs. In this way our encoding allows to bind names: indeed, all the nodes $\circ$ appearing in the interface represent free names while all the others represents bound names.

The graphical encoding presented above is not sound and complete with respect to the structural congruence $\equiv$ presented in Figure 3.2. It is easy to see for example that the processes $(\nu k)(out\ m.open\ k)$ and $out\ m.(\nu k)open\ k$, for $m \neq k$, are mapped to the same graph, represented in Figure 4.7. Therefore, this graphical encoding forces to consider new structural axioms, that is, "floating" axioms for capabilities, concisely represented by the axiom in Figure 4.8.

Moreover, as for the solution proposed in Section 3.4, the soundness of the encoding requires the structural axiom *Cong-Res-Nil* to be dropped. Note that considering the standard structural congruence of mobile ambients with the axioms *Cong-Res-Act* and without the axiom *Cong-Res-Nil* does not change substantially the reduction semantics. The equality introduced by the axioms *Cong-Res-Act* holds for example in the observational equivalence for mobile ambients proposed in [47].

The encoding is sound and complete with respect to the structural congruence $\equiv$, now induced by the axioms in Figure 3.2 and the ones in Figure 4.8, as stated by the proposition below.

Proposition 4.1. Let P, Q be pure processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. Then, $P \equiv Q$ if and only if $[\![P]\!]^{go}_\Gamma = [\![Q]\!]^{go}_\Gamma$.

The proof is very similar to the one for Proposition 3.1. The result could be suitably extended, in order to encompass also well-formed processes, but this is not necessary for our purposes.

4.4 Graph Transformation for the Extended Mobile Ambients

To model the reduction semantics of the extended mobile ambients, we adopt a slight variant of the graph transformation system $\mathcal{R}_{amb}$ presented in Section 3.5 (Figure 3.16).

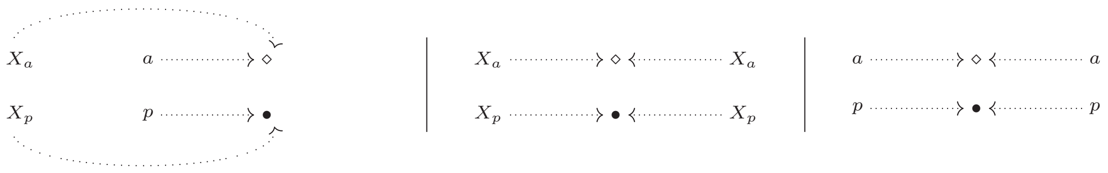


Figure 4.3: Graphs $\mathbf{0}_{X_a}$ and $\mathbf{0}_{X_p}$; id_{X_a} and id_{X_p}; id_a and id_p (top to bottom and left to right).

$$\begin{array}{lcll}
[\![X]\!]_\Gamma & = & \mathbf{0}_X \otimes \mathbf{0}_\Gamma & \\
[\![\mathbf{0}]\!]_\Gamma & = & \mathbf{0}_{a,p} \otimes \mathbf{0}_\Gamma & \\
[\![n[P_\epsilon]]\!]_\Gamma & = & (id_{pv(P_\epsilon)} \otimes amb_n \otimes id_\Gamma) \circ [\![P_\epsilon]\!]_\Gamma & \\
[\![M.P_\epsilon]\!]_\Gamma & = & (id_{pv(P_\epsilon)} \otimes act_n \otimes id_\Gamma) \circ [\![P_\epsilon]\!]_\Gamma & \\
[\![P_\epsilon \mid Q_\epsilon]\!]_\Gamma & = & [\![P_\epsilon]\!]_\Gamma \otimes [\![Q_\epsilon]\!]_\Gamma & \\
[\![(\nu n)P_\epsilon]\!]_\Gamma & = & (id_{pv(P_\epsilon)} \otimes id_{a,p} \otimes new_m \otimes id_\Gamma) \circ [\![P_\epsilon\{^m/_n\}]\!]_{\Gamma \cup \{m\}} & \text{for } m \notin \Gamma
\end{array}$$

Figure 4.4: Encoding for well-formed processes.

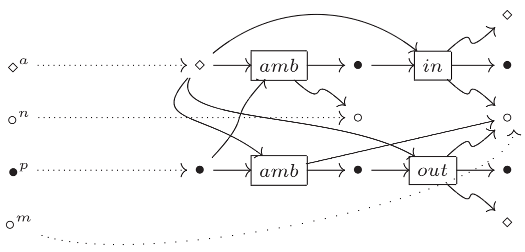


Figure 4.5: Graph encoding for the process $n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]$.

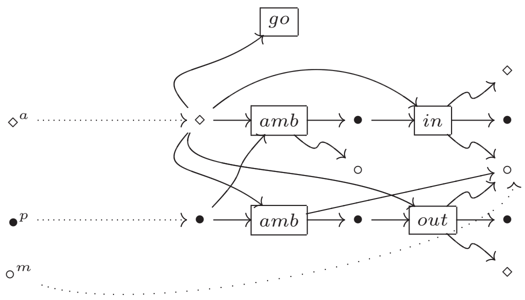


Figure 4.6: Graph encoding for the process $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}])$.

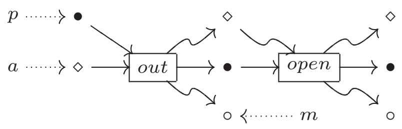


Figure 4.7: Encoding for $(\nu k)(out\ m.open\ k)$ and $out\ m.(\nu k)open\ k$.

$(\nu n)M.P = M.(\nu n)P$ if $n \notin fn(M)$ (Cong-Res-Act)

Figure 4.8: The capability floating axiom.

The new rewriting rules are shown in Figure 4.9. The main difference is that here we have to consider only injective matches, therefore we need to assume an instance for the rules p_{in} and p_{out}, where the nodes labelled n and m may actually be coalesced. Moreover, we consider the rules with the intermediate graph I without edges except the go edge. This allows us to make easier the synthesis of borrowed contexts derivations, by preserving the soundness and completeness of the encoding with respect to the reduction relation $\rightarrow$.

Theorem 4.1 (Reductions vs. rewrites)**.** *Let P be a pure process, and let Γ be a set of names, such that $fn(P) \subseteq \Gamma$. If $P \rightarrow Q$, then $\mathcal{R}_{amb}$ entails a direct derivation $[\![P]\!]^{go}_{\Gamma} \Longrightarrow [\![Q]\!]^{go}_{\Gamma}$. Vive versa, if $\mathcal{R}_{amb}$ entails a direct derivation $[\![P]\!]^{go}_{\Gamma} \Longrightarrow \mathbb{G}$, then there exists a pure process Q, such that $P \rightarrow Q$ and $\mathbb{G} = [\![Q]\!]^{go}_{\Gamma}$.*

The proof is very similar to the ones of Theorems 3.2 and 3.3.

In the following we will introduce an example of application of the p_{in_c} rule. Before presenting it, we would point out that, thanks to the special form of the DPO rules in Figure 4.9, given a match m, the pushout (1) of Definition 4.3 always exists.

Example 4.2. Let T be the pure process S previously introduced in Section 3.2.2 but with the ambient name n coinciding with m, namely, $T = (\nu m)(m[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}])$.

The enriched graphical encoding for the process above is depicted in Figure 4.10. The edge labelled *go* denotes the entry point for the computation of the process. Note that all the edges of the graph can be involved in a reduction step because they have the same activation node with an outgoing go edge.

The application of the p_{in-c} rule to the graph in Figure 4.10 results in the graph in Figure 4.11, which is the encoding for the process $(\nu m)(m[m[\mathbf{0}]|out\ m.\mathbf{0}])$. This rewriting step simulates the transition $T \rightarrow (\nu m)(m[m[\mathbf{0}]|out\ m.\mathbf{0}])$. With respect to $m[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}] \rightarrow m[m[\mathbf{0}]|out\ m.\mathbf{0}]$, the application of the (ResRed) rule is immaterial: the occurrence of the restriction operator is simply mimicked by the embedding of L_{in-c} into a graph with an interface that is lacking m.

4.5 The Synthesized Transition System

In this section we start applying the BC synthesis mechanism to $\mathcal{R}_{amb}$ in order to derive an LTS for graphs representing mobile ambients processes. We open with an introductory section explaining the graphical counterpart of process variables (Section 4.5.1): these are employed in the presentation (Section 4.5.2) of some examples of rewriting steps with BCs. Building on these, we then introduce (Section 4.5.4) a compact representation of the derived LTS by means of *minimal derivations*: these are extrapolated via the use of some pruning techniques (Section 4.5.3). The resulting LTS is going to be exploited in Section 4.6, in order to define a novel LTS directly for mobile ambients processes.

4.5.1 Process variables, graphically

We first illustrate how a single BC transition may induce a reduction involving extended processes. To this end, consider the graph $J \rightarrowtail G$ depicted in Figure 4.12 and the diagram in Definition 2.20. The former represents the encoding of the process $S = (\nu n)(m[\mathbf{0}] \mid n[\mathbf{0}])$.

The occurrence of the nodes $\bullet^{1_p}$ and $\diamond^{1_a}$ ensures us that the process represented by $J \rightarrowtail F$, namely $T = open\, m.\mathbf{0}$, can be put in parallel with S, so that $J \rightarrowtail G^+$ intuitively corresponds to $S \mid T$. Note however the occurrence of the nodes $\bullet^{2_p}$ and $\diamond^{2_a}$ in K: they witness the possibility of a parametric instance of process T. Indeed, the graph with interfaces $K \rightarrowtail G^+$ actually represents $S \mid T_X$, for any process variable X and well-formed process $T_X = open\, m.X$.

Put differently, the context $J \rightarrowtail F \leftarrowtail K$ is the minimal context allowing the reduction, which can be obtained by applying the BC technique. The presence of the nodes $\bullet^{2_p}$ and $\diamond^{2_a}$ in K is important because they denote the fact that it could be further instantiated with any substitution of the process variable X.

Additionally, note why our composition does not capture bound names. Consider e.g. the bound name n of $G \leftarrowtail J$. It does not appear in the interface J and thus, for all graph with interfaces $J \leftarrowtail F' \rightarrowtail K'$ (representing possible substitutions), it can not be identified with any name of F'.

4.5.2 Examples of borrowed transitions

This section shows the application of the BC synthesis mechanism to the graphical encoding of a process. Let us consider the graph $J \rightarrowtail G = [\![P]\!]^{go}_{\{m\}}$, where $P = (\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}])$. In the following

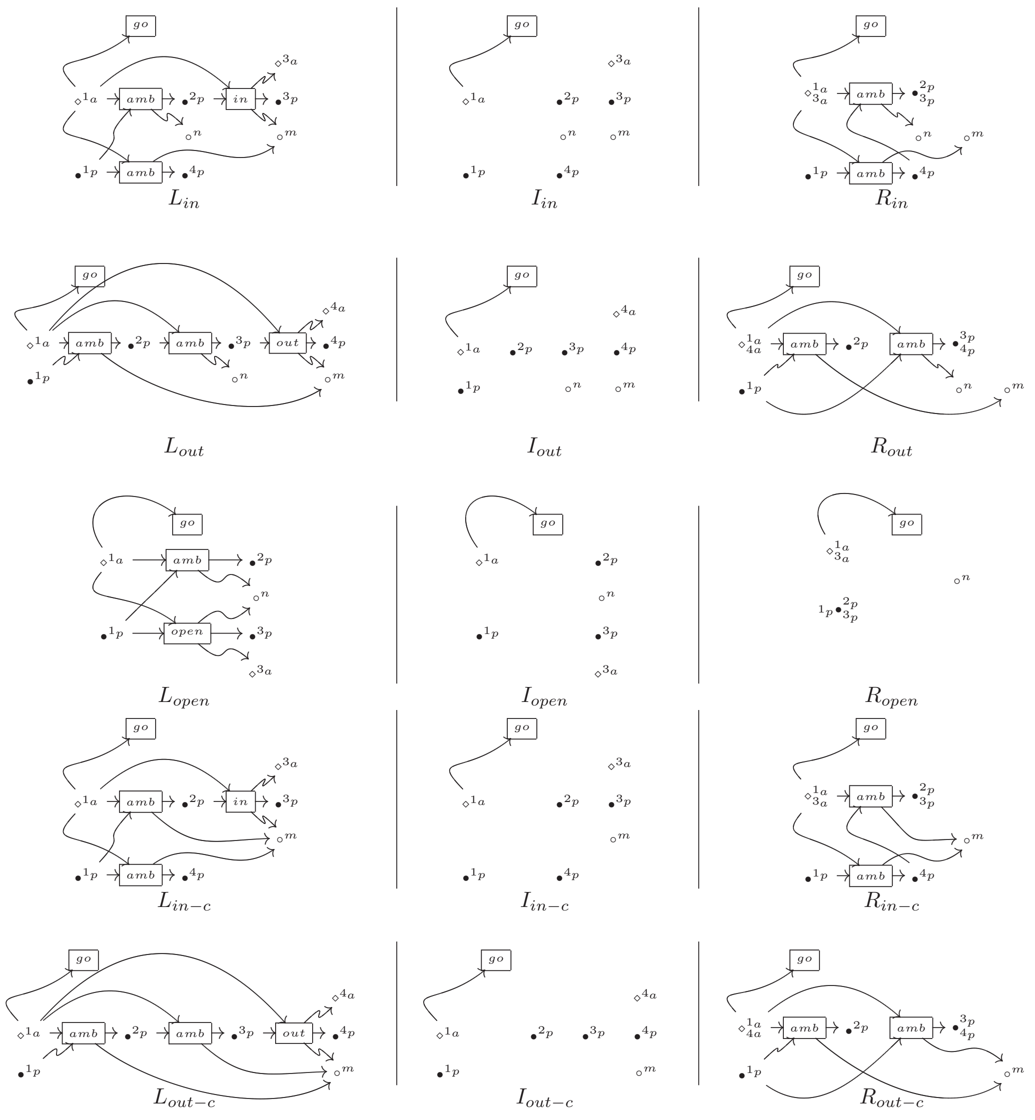


Figure 4.9: The rewriting rules p_{in}, p_{out}, p_{open}, p_{in-c} and p_{out-c} (top to bottom).

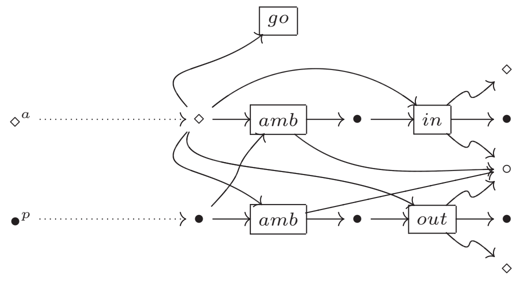


Figure 4.10: Graph encoding for the process $(\nu m)(m[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}])$.

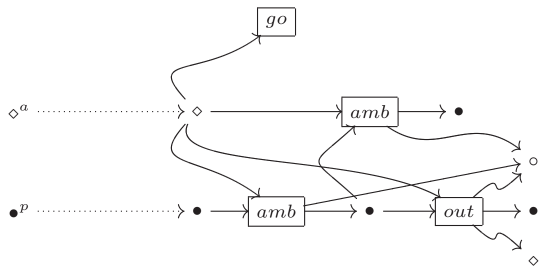


Figure 4.11: Graph encoding for the process $(\nu m)(m[m[\mathbf{0}]|out\ m.\mathbf{0}])$.

we discuss the possible transitions with source $J \rightarrowtail G$ that are induced by the rule $p_{in} : L_{in} \leftarrowtail I_{in} \rightarrow R_{in}$ of $\mathcal{R}_{amb}$ in Figure 4.9. Since for each pair of monos $G \leftarrowtail D \rightarrowtail L_{in}$ a labelled transition might exist, in order to perform a complete analysis, we should consider all the pairs of monos $G \leftarrowtail D \rightarrowtail L_{in}$. We proceed by showing some of the possible transitions generated by such pairs. Actually, we are going to see that it is not necessary to check all those pairs that we are not considering here, by exploiting the pruning techniques presented in the next subsection.

BC transition for D equals to L_{in} Let us take as D the left-hand side L_{in} and note that there is only one map into the graph G. The transition generated by this choice is depicted in Figure 4.22. The graph G^+ is the same as G. Now C and H are constructed as in a standard DPO rewriting step. When taking D as the whole left-hand side, $J \rightarrowtail G$ needs no context for the reduction and thus the label of this transition is the identity context, i.e., two isomorphisms into the discrete graphs with three nodes $\{p, a, m\}$.[1] Intuitively, this corresponds to an internal transition over processes, labelled with τ.

BC transition for D equals to the subgraph in the upper part of L_{in} Now we take as D the subgraph of L_{in} representing an ambient with a capability in inside it. Note that also in this case there is only one possible map into the graph G. The resulting transition is shown in Figure 4.23. The graph G^+ is the graph G in parallel with the graph representing an ambient m, thus intuitively it represents the process $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]|m[X])$ for some process variable X. The graph $J \rightarrowtail G$, in order to reach the graph G^+, has to borrow from the environment the context $J \rightarrowtail F \leftarrowtail K$ that represents the syntactic context $-|m[X]$. Note that in the resulting interface K there is a process node $\bullet^{4p}$ pointing to the process node of F occurring inside the ambient m, and this process node represents a process variable X, as detailed in Section 4.5.1. The graphs C and H are then constructed as in the standard DPO approach. Intuitively, $K \rightarrow H$ represents the process $m[out\ m.\mathbf{0}]|m[n[\mathbf{0}]|X]$, where X is the same process variable occurring in the label $J \rightarrowtail F \leftarrowtail K$. This can be understood by observing that the process node $\bullet^{4p}$ of K points both to a node of H and to a node of F. Summarizing, this transition moves the ambient n into an ambient m that is provided by the environment.

[1]Or, equivalently, to the value of the expression $id_p \otimes id_a \otimes id_m$, as defined in Section 3.4.

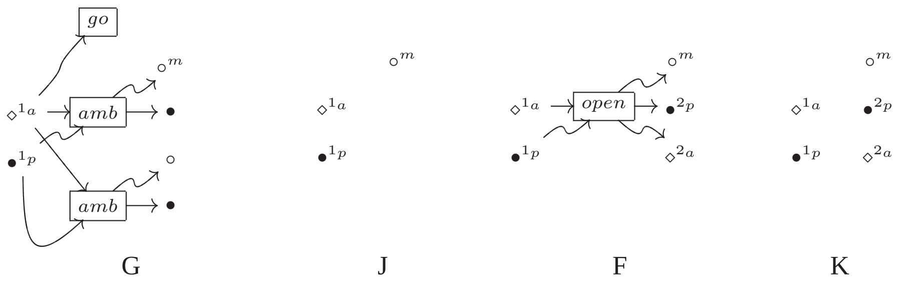


Figure 4.12: The graphs with interfaces $J \rightarrowtail G$ and the context $J \rightarrowtail F \leftarrowtail K$.

BC transition for D equals to the subgraph in the lower part of L_{in} Another possible D is the subgraph of L_{in} consisting of the ambient depicted in the lower part of L_{in}. In this case, there are two possible maps into the graph G: the map into the subgraph of G representing the ambient m, and the map into the subgraph of G representing the restricted ambient n.

In the first case, we obtain the transition shown in Figure 4.24. The graph G^+ is the graph G in parallel with the graph representing a fresh ambient name w having inside a capability $in\ m$. Intuitively, it represents the process $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]|w[in\ m.X_2|X_1])$ for some process variables X_1, X_2. In order to reach G^+, the graph $J \rightarrowtail G$ has to borrow from the environment the context $J \rightarrowtail F \leftarrowtail K$ representing the syntactic context $-|w[in\ m.X_2|X_1]$. As in the above case X_1 and X_2 are process variables, since in the interface K there are the process nodes $\bullet^{2_p}$ and $\bullet^{3_p}$. The graphs C and H are obtained by a standard DPO derivation. The graph $K \rightarrow H$ represents the process $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}|w[X_2|X_1]])$. Summarizing, this transition represents an ambient w from the environment entering inside the ambient m of the process P.

In the second case no transition is possible. Indeed the graph G^+ is the whole graph G in parallel with a fresh ambient w having inside a capability $in\ n$, but the pushout complement of $J \rightarrowtail G \rightarrowtail G^+$ does not exist, because n is restricted and thus it does not belong to the interface J. Intuitively, this means that no ambient from the environment can enter inside a restricted sibling ambient n.

4.5.3 Reducing the Borrowing

In order to know all the possible transitions originating from a graph with interfaces $J \rightarrowtail G$, all the subgraphs D's of L_{in}, L_{out} and L_{open} should be analyzed. To shorten this long and tedious procedure, we use the two pruning techniques presented in [7].

The first one is based on the observation that those items of a left-hand side L that are not in D have to be glued to G through J. Let us consider a node n of D corresponding to a node n' in L, such that n' is the source or the target of some edge e that does not occur in D. Since the edge e is in L but not in D, it must be added to G through J, and thus n, must be also in J. Such nodes are called *boundary node*.

Let us consider for example the graph in Figure 4.13 as a subgraph of L_{open}. Its root nodes are boundary nodes since they have an outgoing edge that occurs in L_{open} but not in it. Also the name node $\circ$ is a boundary node, since in L_{open} there is an ingoing edge that does not occur in the graph in Figure 4.13. Therefore these nodes must be mapped to nodes occurring in the interface J of G. This is exactly the reason why, if we consider the graph $J \rightarrowtail G$ in Figure 4.12 there is a transition when we choose as D the graph in Figure 4.13 mapped to the subgraph representing the ambient m, while no transition is possible if we map the same D to the subgraph modelling the ambient n.

Boundary nodes are formally captured by the categorical notion of *initial pushout*.

Definition 4.5 (Initial pushout)**.** *Let the square (1) below be a pushout. It is an* initial pushout *of $C \rightarrow D$ if for every other pushout as in diagram (2) there exist two unique morphisms $A \rightarrow A'$ and $B \rightarrow B'$ such that diagram (2) commutes.*

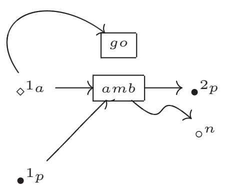


Figure 4.13: The subgraph of L_{open}.

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & B \arrow[d] \\
C \arrow[r] & D
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[rrr] \arrow[ddd] \arrow[dr, dotted] & & & B \arrow[ddd] \arrow[dl, dotted] \\
& A' \arrow[r] \arrow[ddl] & B' \arrow[ddr] & \\
& & PO & \\
C \arrow[rrr] & & & D
\end{tikzcd}$$

(1) (2)

Since the category of (typed hyper-)graphs we work in has initial pushouts for all arrows [25], the previous discussion is formalized by the lemma below [7, Corollary 1].

Lemma 4.1. *A graph with interfaces $J \to G$ can perform a BC rewriting step in $\mathcal{R}_{amb}$ if and only if there exist*

- *a mono $D \rightarrowtail L$ (where L is the left hand side of some production in $\mathcal{R}_{amb}$),*
- *a mono $D \rightarrowtail G$,*
- *a morphism $J_D \to J$ (where J_D is the initial pushout of $D \rightarrowtail L$) such that square (2) in Figure 4.14 commutes.*

The three conditions of the lemma above are sufficient to guarantee that a graph $J \to G$ can perform a BC rewriting step. This is indeed possible if and only if there exist a mono $D \rightarrowtail G$ and a mono $D \rightarrowtail L$ such that the diagram of Definition 2.20 can be constructed. Since pushouts and pullbacks always exist, we have just to ensure that pushout complements exist. Now, as said in Section 4.4, for all the rules in Figure 4.9, the pushout complement $I \rightarrowtail L \rightarrowtail G^+$ always exists because all the nodes of L are in I. Thus, we have a transition if and only if there exists the pushout complement $J \to G \rightarrowtail G^+$ which, as shown in [7], is guaranteed by the third condition of the lemma.

This lemma allows to heavily prune the space of possible D's. As for graphs corresponding to the encoding of processes, we can exclude all those D's having a continuation process node (any node depicted by $\bullet$ that is not the root) as boundary node, observing that the only process node in the interface J is the root node.

A further pruning —partially based on proof techniques presented in [28]— is performed by excluding all those D's which generate a BC transition that is not relevant for the bisimilarity. In general terms, we may exclude all the D's that contain only nodes, since those D's can be embedded in every graph (with the same interface) generating the same transitions.

Concerning our case study, those transitions generated by a D having the root node without the edge labelled go are also not relevant. In fact, a graph can perform a BC transition using such a D if and only if it can perform a transition using the same D with a go edge outgoing from the root. Note indeed that the resulting states of these two transitions only differ for the number of go edges attached to the root: the state resulting after the first transition has two go's, the state resulting after the second transition only one. These states are bisimilar, since the number of go's does not change the behavior [7, Lemma 12].

$$\begin{tikzcd}
J_D \arrow[r] \arrow[d] \arrow[dr, phantom, "(1)"] \arrow[ddd, bend right=40] & F_D \arrow[d] & & \\
D \arrow[r, tail] \arrow[d, tail, "(2)"'] \arrow[dr, phantom, "PO"] & L \arrow[d] \arrow[dr, phantom, "PO"] & I \arrow[l, tail] \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "PO"] & R \arrow[d, tail] \\
G \arrow[r, tail] \arrow[dr, phantom, "PO"] & G^+ \arrow[dr, phantom, "PB"] & C \arrow[l, tail] \arrow[r] & H \\
J \arrow[r, tail] \arrow[u] & F \arrow[u] & K \arrow[l, tail] \arrow[u] \arrow[ur, dotted] &
\end{tikzcd}$$

Figure 4.14: The BC construction together with commuting squares (1) (the initial pushout of $D \rightarrowtail L$) and (2).

The two pruning techniques presented above allow us to only consider the partial matches D shown in Figures 4.15, 4.25 and 4.26, together with D's obtained from the ones of the last two figures by coalescing the name nodes n and m.

4.5.4 Minimal transitions

In Section 4.5.3 we restricted quite a lot the space of possible D's. However, reasoning on the synthesized LTS is still hard (this is usually the case when working with derived LTSs, as pointed out in [3] and [6], where the authors state that an SOS presentation of the synthesized LTS would be desirable). In order to simplify this reasoning, we introduce a set of *minimal transitions* that allow us to derive all and only the transitions of the (pruned) synthesized LTS.

Inspired by Lemma 4.1, providing necessary and sufficient conditions for performing a transition, we consider the graphs $J_D \to D$ for all those D's that have not been pruned in Section 4.5.3 and J_D containing only the boundary nodes of D.

The minimal transitions have the following shape where the leftmost square in the lower row is the initial pushout of $D \rightarrowtail L$.

$$\begin{tikzcd}
D \arrow[r, tail] \arrow[d] \arrow[dr, phantom, "PO"] & L \arrow[d] \arrow[dr, phantom, "PO"] & I \arrow[l, tail] \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & R \arrow[d] \\
D \arrow[r, tail] & L & I \arrow[l, tail] \arrow[r] & R \\
J_D \arrow[r, tail] \arrow[u] \arrow[ur, phantom, "IPO"] & F_D \arrow[u] \arrow[ur, phantom, "PB"] & K_D \arrow[l, tail] \arrow[u] \arrow[ur, dotted]
\end{tikzcd}$$

Figures 4.15, 4.25 and 4.26 concisely represent these transitions, showing for each of these the starting graph D, the label $J_D \rightarrowtail F_D \leftarrowtail K_D$, and the resulting graph R. The three figures represent the minimal transitions respectively generated by the rules p_{open}, p_{in} and p_{out}. Additionally, the minimal transitions generated by the rules p_{in-c} and p_{out-c} should be considered, but they are easily described starting from those of p_{in} and p_{out}, respectively. In particular, for each minimal transition with D_{in_x} there exists a minimal transition generated by p_{in-c}, where all the relevant graphs are obtained by coalescing the nodes n and m.[2] Analogously for the minimal transitions generated by the rule p'_{out}.

All the transitions originated from a graph $J \rightarrowtail G$ (representing a process) can be characterized by exploiting these minimal transitions. By Lemma 4.1, we can state that $J \rightarrowtail G$ can perform a BC rewriting step in $\mathcal{R}_{amb}$ if and only if there exist a mono $D \rightarrowtail G$, for some D of the minimal transitions, and a morphism $J_D \to J$ such that square (2) in Figure 4.14 commutes.

The label of the rewriting step can be obtained from the label of the minimal transition. First of all note that the interface J contains all the nodes of J_D (as suggested by the morphism $J_D \to J$), all the name nodes $\circ$ representing the free names of the modeled process (as expected by our encoding), and the root nodes of the graph D when they are not in J_D. Then the graph F only contains the whole graph F_D and all the nodes of J. Indeed, as shown in the proposition below, which is an adaptation of Proposition 4 of [7], F can be obtained as the pushout of $J_D \to F_D$ and $J_D \to J$.

Proposition 4.2. Let $p : L \leftarrowtail I \to R$ be a production of $\mathcal{R}_{amb}$; $d : D \rightarrowtail L$ a mono such that in Figure 4.16 diagram (i) is the initial pushout of d and diagram (ii) is a pullback; and $J \rightarrowtail G$ a graph with interfaces. Then there exists a K such that $J \rightarrowtail G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H$ *via* p and d if and only if there exists a mono $D \rightarrowtail G$, a graph V and a morphism $J_D \to J$ such that the central square of diagram (iii) in Figure 4.16 commutes and F and H are constructed as illustrated there.

It is easy to prove that K is a discrete graph containing all and only the nodes of F, or more concretely, K consists of the nodes of J and K_D.

Finally, the resulting graph H is obtained by replacing in the graph G the subgraph D with R. As shown in Proposition 4.2, it can be computed in a DPO step of $D \leftarrowtail D \cap I \to R$, where $D \cap I$ is the pullback of $D \rightarrowtail L$ and $I \rightarrowtail L$.

[2]Note also that it is irrelevant to consider the coalesced version for the rule with D'_{in_i}, since it would coincide with the minimal transition for D_{in-c_i}, for all i.

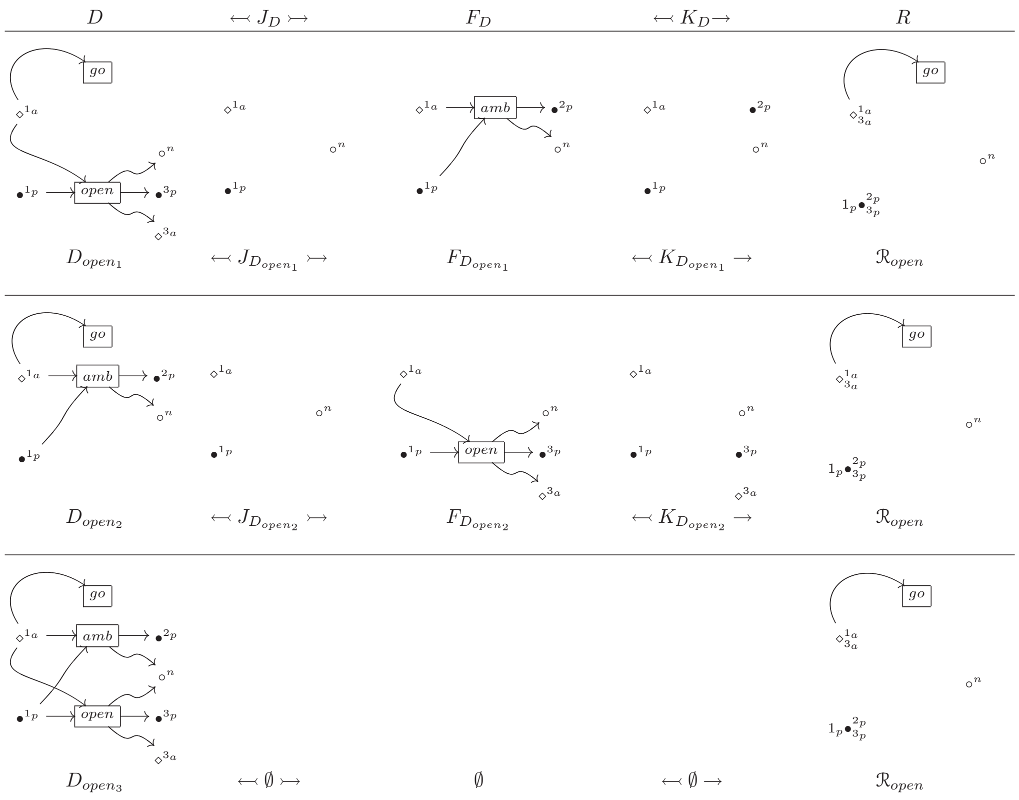

D
$\leftarrowtail J_D \rightarrowtail$
F_D
$\leftarrowtail K_D \rightarrow$
R
D_{open_1}
$\leftarrowtail J_{D_{open_1}} \rightarrowtail$
$F_{D_{open_1}}$
$\leftarrowtail K_{D_{open_1}} \rightarrow$
$\mathcal{R}_{open}$
D_{open_2}
$\leftarrowtail J_{D_{open_2}} \rightarrowtail$
$F_{D_{open_2}}$
$\leftarrowtail K_{D_{open_2}} \rightarrow$
$\mathcal{R}_{open}$
D_{open_3}
$\leftarrowtail \emptyset \rightarrowtail$
$\emptyset$
$\leftarrowtail \emptyset \rightarrow$
$\mathcal{R}_{open}$
go
amb
open

Figure 4.15: The minimal transitions generated by the rule p_{open}.

As an example, consider the BC rewriting step shown in Figure 4.22. We are going to show that it is derivable by the minimal transition for D_{in_4}, shown in Figure 4.25. First of all note that there exist $D_{in_4} \to G$ and $\emptyset \to J$ such that the square (2) in Figure 4.14 commutes. Now, F is equal to J, since it consists of the composition of $F_{D_{in_4}}$ (i.e., $\emptyset$) and J. The new interface K is equal to F, since it contains all and only the nodes of J and $K_{D_{in_4}}$ (i.e., $\emptyset$). The arriving state H is obtained simply by replacing D_{in_4} with R_{in}. Therefore, starting from the minimal transition D_{in_4}, we exactly obtained the BC transition of Figure 4.22.

4.6 A New LTS for Mobile Ambients

This section presents the LTS $\mathcal{D}$ directly defined over mobile ambients processes. The inference rules describing this LTS are obtained from the transitions of the LTS on graphs presented in Section 4.5.4. In particular, we derive an inference rule for each minimal transition. As we will explain later in Section 4.6.2, the conditions in the premise of each inference rule correspond to the necessary and sufficient conditions allowing a transition from a graph G encoding a process, while the label and the resulting process are obtained from the label and the resulting state of the borrowed transition, respectively. Section 4.6.1 presents the LTS, while Section 4.6.2 shows how this LTS is distilled starting from the LTS over graphs.

The labels of the transitions are unary contexts, i.e., terms of the extended syntax with a hole $-$. The formal definition of our LTS is shown in Figures 4.17 and 4.18.

4.6.1 The labelled rules on processes...

The rules in Figure 4.17 represent the τ-actions modeling internal computations. Note that the labels of the transitions are contexts composed of just a hole $-$, while the resulting states are pure processes. The rule InTau enables an ambient n to enter a sibling ambient m. The rule OutTau enables an ambient n to get out of its parent ambient m. Finally, the rule OpenTau models the opening of an ambient n. These three rules exactly derive the same transition relation of the reduction relation over mobile ambients, thus they could be replaced with the rules in Figure 3.32.

The rules in Figure 4.18 model the interactions of a process with its environment. Note that both labels and resulting states contain process and name variables. We define the LTS $\mathcal{D}_\mathcal{I}$ for pure processes of mobile ambients by instantiating all the variables of the labels and of the resulting states. We implicitly assume that it is closed with respect to the structural congruence.

Definition 4.6. *Let P, Q be pure processes and let $C[-]$ be a pure context. Then, we have that $P \xrightarrow{C[-]}_{\mathcal{D}_\mathcal{I}} Q$ if there exists a transition $P \xrightarrow{C_\epsilon[-]}_{\mathcal{D}} Q_\epsilon$ and a substitution σ such that $Q_\epsilon\sigma = Q$ and $C_\epsilon[-]\sigma = C[-]$.*

Recall that substitutions map process variables into pure processes, and that they do not capture bound names.

The rule Open models the opening of an ambient provided by the environment. In particular, it enables a process P with a capability $open\ n.P_1$ at top level, for $n \in fn(P)$, to interact with a context providing an ambient n that contains inside it some process X_1. The resulting state is the process over the extended syntax $(\nu A)(P_1|X_1|P_2)$, where X_1 represents a process provided by the environment. Note that the instantiation of the process variable X_1 with a process containing a free name that belongs to the bound names in A is possible only α-converting the resulting process $(\nu A)(P_1|X_1|P_2)$ into a process that does not contain that name among its bound names at top level.

$$\begin{tikzcd}
J_D \arrow[r] \arrow[d] \arrow[dr, phantom, "IPO"] & F_D \arrow[d] \\
D \arrow[r, tail] & L
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
D \arrow[d, tail] \arrow[dr, phantom, "PB"] & D \cap I \arrow[l, tail] \arrow[d, tail] \\
L & I \arrow[l, tail]
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
F_D \arrow[d] \arrow[dr, phantom, "PO"] & J_D \arrow[l, tail] \arrow[r, tail] \arrow[d, tail] \arrow[dr, phantom, "="] & D \arrow[d, tail] \arrow[dr, phantom, "PO"] & D \cap I \arrow[l, tail] \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "PO"] & R \arrow[d, tail] \\
F & J \arrow[l, tail] \arrow[r, tail] \arrow[rr, tail, dotted, bend right] & G & V \arrow[l, tail] \arrow[r] & H
\end{tikzcd}$$

(iii)

Figure 4.16: Diagrams used in Proposition 4.2.

$$(\textsc{InTau})\quad \frac{P\equiv(\nu A)\ \mathcal{C}[n[in\ m.P_1|P_2]|m[P_3]]}{P\xrightarrow{-}(\nu A)\ \mathcal{C}[m[n[P_1|P_2]|P_3]]} \qquad (\textsc{OutTau})\quad \frac{P\equiv(\nu A)\ \mathcal{C}[m[n[out\ m.P_1|P_2]|P_3]]}{P\xrightarrow{-}(\nu A)\ \mathcal{C}[m[P_3]|n[P_1|P_2]]}$$

$$(\textsc{OpenTau})\quad \frac{P\equiv(\nu A)\ \mathcal{C}[n[P_1]|open\ n.P_2]}{P\xrightarrow{-}(\nu A)\ \mathcal{C}[P_1|P_2]}$$

Figure 4.17: The internal transitions of the LTS $\mathcal{D}$ (for $\mathcal{C}[-]$ context containing only ambients and parallel operators).

$$(\textsc{In})\quad \frac{P\equiv(\nu A)(in\ m.P_1|P_2)\quad m\notin A}{P\xrightarrow{x[-|X_1]|m[X_2]}(\nu A)m[x[P_1|P_2|X_1]|X_2]} \qquad (\textsc{OutAmb})\quad \frac{P\equiv(\nu A)(n[out\ m.P_1|P_2]|P_3)\quad m\notin A}{P\xrightarrow{m[-|X_2]}(\nu A)(m[P_3|X_2]|n[P_1|P_2])}$$

$$(\textsc{InAmb})\quad \frac{P\equiv(\nu A)(n[in\ m.P_1|P_2]|P_3)\quad m\notin A}{P\xrightarrow{-|m[X_2]}(\nu A)(m[n[P_1|P_2]|X_2]|P_3)} \qquad (\textsc{Open})\quad \frac{P\equiv(\nu A)(open\ n.P_1|P_2)\quad n\notin A}{P\xrightarrow{-|n[X_1]}(\nu A)(P_1|X_1|P_2)}$$

$$(\textsc{CoIn})\quad \frac{P\equiv(\nu A)(m[P_1]|P_2)\quad m\notin A}{P\xrightarrow{-|x[in\ m.X_1|X_2]}(\nu A)(m[x[X_1|X_2]|P_1]|P_2)} \qquad (\textsc{CoOpen})\quad \frac{P\equiv(\nu A)(n[P_1]|P_2)\quad n\notin A}{P\xrightarrow{-|open\ n.X_1}(\nu A)(P_1|X_1|P_2)}$$

$$(\textsc{Out})\quad \frac{P\equiv(\nu A)(out\ m.P_1|P_2)\quad m\notin A}{P\xrightarrow{m[x[-|X_1]|X_2]}(\nu A)(m[X_2]|x[P_1|P_2|X_1])}$$

Figure 4.18: The environmental transitions of the LTS $\mathcal{D}$.

The rule CoOpen instead models an environment that opens an ambient of the process. The rule InAmb enables an ambient of the process to migrate into a sibling ambient provided by the environment, while in the rule In both the ambients are provided by the environment. In the rule CoIn an ambient provided by the environment enters an ambient of the process. The rule OutAmb models an ambient of the process exiting from an ambient provided by the environment, while in the rule Out both ambients are provided by the environment.

The LTS $\mathcal{D}$ does not properly conform to the so-called SOS style: indeed, the premises of the inference rules are just constraints over the structure of the process. This is a consequence of the fact that the rules of our LTS are obtained from the borrowed minimal transitions. Each rule corresponds to one minimal transition presented in Section 4.5.4 and it is obtained as described below.

4.6.2 ...from the borrowed rules on graphs

Observe that a graph $J \rightarrowtail G$ representing a process P can perform a BC rewriting step in $\mathcal{R}_{amb}$ if and only if there exists a mono $D \rightarrowtail G$, for some D of a minimal transition, and a morphism $J_D \rightarrow J$, such that square (2) in Figure 4.14 commutes. Moreover, the label and the resulting graph of the borrowed transition for G are obtained from the label and the resulting state of the minimal transition of D, respectively. Therefore, for each minimal transition we obtain an inference rule: the conditions in the premise correspond to the necessary and sufficient conditions for performing a transition from a graph G, while the label and the resulting process are obtained from the label and the resulting state of the borrowed transition, respectively. Since the labels of the LTS over graphs obtained by the BC mechanism represent minimal graph contexts enabling a graph production, the labels of our LTS over processes represent minimal process contexts enabling a reduction.

As the main example, in this section we closely look at the correspondence between the rule Open and the first minimal transition in Figure 4.15.

Consider a graph $J \rightarrowtail G$ representing the encoding for a process P. If there exists a mono $D_{open_1} \rightarrowtail G$ and a morphism $J_{D_{open_1}} \rightarrow J$, such that the square (2) in Figure 4.14 commutes, the graph $J \rightarrowtail G$ can perform a BC rewriting step in $\mathcal{R}_{amb}$ with label $J \rightarrowtail F \leftarrowtail K$, where J, F and K respectively consist of $J_{D_{open_1}}$, $F_{D_{open_1}}$ and $K_{D_{open_1}}$ together with the free names of P. Now, note that D_{open_1} can be embedded in G and a morphism $J_{D_{open_1}} \rightarrow J$ (such that the square (2) in Figure 4.14 commutes) may exist if and only if $P \equiv (\nu A)(open\ n.P_1|P_2)$, for $n \notin A$. Indeed, the graph must contain an occurrence of the operator

$open\ n.-$ on top, possibly further instantiated, since it includes D_{open_1}; and since the interface J contains all the nodes of $J_{D_{open_1}}$, we conclude that n must belong to J, that is, n must be a free name of P. This is the premise of the rule OPEN.

Starting from the label $J \rightarrowtail F \leftarrowtail K$ of the BC transition we now obtain the label of the process transition. By observing the shape of F, which contains all the items of $F_{D_{open_1}}$, we can say that the process context is composed of the ambient n. Moreover, the context F is glued to G through J, which contains the free names of P and the nodes of $J_{D_{open_1}}$, i.e., the name n and the nodes representing the roots of the graph G (which models P). Since these two nodes represent the roots of the graph F (which models ambient n), we conclude that the label of the process transition is a context with the ambient n in parallel with a hole representing process P.

The graph K represents the interface of both graphs F and H. It contains all the nodes of $K_{D_{open_1}}$, i.e., the roots of F and the roots of the process inside the ambient n. The nodes of the interface K represent the "handles" of F and H for interacting with an environment. Therefore, the process node of K that is not the root of F can be thought of as a process variable inside the ambient n in the label of the transition. Therefore, we conclude that the label of the transition with source the process P can be represented as the minimal context $-|n[X_1]$, where $-$ is a hole and X_1 is a process variable. The resulting process $(\nu A)(P_1|X_1|P_2)$ exactly corresponds to the state H from the BC transition. Indeed, in the interface K of the graph $K \to H$ also the node modeling the process variable X_1 occurs, which represents a process provided by the environment.

The reader should notice that while there are 13 minimal transitions, only 10 rules occur in Figures 4.17 and 4.18. This is due to the fact that each of the rules IN, COIN and OUT is actually derived by two minimal transitions. The rule IN is generated by the minimal transitions D_{in_1} and D'_{in_1} of Figure 4.25, COIN by D_{in_3} and D'_{in_3} of the same figure, and OUT by D_{out_1} and D'_{out_1} of Figure 4.26. We show the latter, since the others are analogous.

In the minimal transition with D_{out_1} two ambients are borrowed from the environment. The first one has name m (i.e., the ambient from which the process wants to exit), while the second has a fresh name n (it is not restricted, since it occurs in $K_{D_{out_1}}$). This transition thus corresponds to the rule

$$\frac{P \equiv (\nu A)(out\ m.P_1|P_2) \quad m \notin A \quad n \notin (A \cup fn(P))}{P \xrightarrow{m[n[-|X_1]|X_2]} (\nu A)(m[X_2]|n[P_1|P_2|X_1])}$$

In the minimal transition with D'_{out_1} the name n belongs to the process (it occurs inside the graph $D_{out'_1}$) but, since the node n occur in $J_{D'_{out_1}}$, it should appear in the interface J, i.e., it must be free. Thus, this transition corresponds to the rule

$$\frac{P \equiv (\nu A)(out\ m.P_1|P_2) \quad m \notin A \quad n \in fn(P)}{P \xrightarrow{m[n[-|X_1]|X_2]} (\nu A)(m[X_2]|n[P_1|P_2|X_1])}$$

The conclusion of the two rules above is identical, thus we can put together their premises, and compactly represent them via the rule OUT of Figure 4.18. Substituting the name n with a name variable x basically guarantees that any actual name can be substituted to n, even m (thanks to D_{out-c_1}), as long as it does not occur in A.

4.7 An SOS Presentation for the Derived LTS $\mathcal{D}$

In the previous two sections we described a semi-automatic methodology for distilling an LTS $\mathcal{D}$. This section introduces a set of SOS rules, tailored over $\mathcal{D}$, such that the associated LTS $\mathcal{S}$ coincides with the former one. The rules for $\mathcal{S}$ are shown in Figure 4.19. We assume the implicit presence of the rule

$$\frac{P \equiv P' \ P' \xrightarrow{C_\epsilon[-]} Q_\epsilon}{P \xrightarrow{C_\epsilon[-]} Q_\epsilon} \qquad \text{(Cong).}$$

The rules in the first two rows of Figure 4.19 model internal computations. They are indeed obtained from the rules in Figure 4.17. In particular, since these rules exactly derive the same transition relation of the reduction relation over mobile ambients, we replace it with the reduction rules labelled with the identity context $-$. So, we obtain the axioms modelling the execution of the capabilities of the calculus, and a structural rule for each ambient, parallel and restriction operators.

The remaining rules in Figure 4.19, modelling the interactions of a process with its environment, are obtained from the rules in Figure 4.18. In particular, for each of these rules we derive three rules. First, we determine the axiom by considering the minimal process needed by the reduction to occur. For e.g. the rule IN of the LTS $\mathcal{D}$, the minimal process allowing the reduction is $in\ m.P_1$. Therefore, we determine the axiom $in\ m.P_1 \xrightarrow{x[-|X_1]|m[X_2]} m[x[P_1|X_1]|X_2]$. The next step is to determine the structural rules in SOS style. So, as far as the rule IN of the LTS $\mathcal{D}$ is concerned, we have that if $P \xrightarrow{x[-|X_1]|m[X_2]} P_\epsilon$, then for the process $P|Q$ there is a transition labelled $x[-|X_1]|m[X_2]$ leading to the process P_ϵ with the process Q inside the ambient x, that is, $P|Q \xrightarrow{x[-|X_1]|m[X_2]} P_\epsilon\{^{Q|X_1}/_{X_1}\}$. Instead, if $P \xrightarrow{x[-|X_1]|m[X_2]} P_\epsilon$ and $m \neq a$, then $(\nu a)P \xrightarrow{x[-|X_1]|m[X_2]} (\nu a)P_\epsilon$.

This result is also confirmed by the analysis of the minimal transitions.

Deriving axioms As explained in Section 4.5.4, a minimal transition represents a BC transition, where the starting graph is the smallest graph allowing a BC rewriting by considering a given graph transformation rule and a given partial match. The graph D of a minimal transition therefore represents the minimal process needed to the reduction modeled by the BC transition to occur. This means that each minimal transition represents an axiom of the SOS LTS.

Let us consider for example the minimal transition for D_{in_1}. The graphs D_{in_1} represents the process $in\ m.\mathbf{0}$, but all the remarks made below also hold for the extended process $in\ m.P_1$, where P_1 represents any process.

As explained in Section 4.5.4, starting from the label of the BC transition we obtain the label of the process transition that in this case is $x[-|X_1]|m[X_2]$, with x name variable. The resulting process is instead represented by the graph R_{in} that models the process $x[P_1|X_1]|m[X_2]$. Therefore, this minimal transition represents the axiom $in\ m.P_1 \xrightarrow{x[-|X_1]|m[X_2]} m[x[P_1|X_1]|X_2]$.

Now, let us consider the minimal transition for D_{in_2}. It represents the axiom $n[(\nu A)(in\ m.P_1|P_2)] \xrightarrow{-|m[X_2]} m[n[(\nu A)(P_1|P_2)]|X_2]$. Nevertheless, it is obvious that this rule can be rewritten as the rule INAMB of Figure 4.19, by using the transition derived according to the rules IN, INPAR and INRES. Graphically, this is suggested by the fact that the graph D_{in_2} contains the partial match D_{in_1}, which gives rise to the minimal transition allowing us to derive the rule of the third row of Figure 4.19.

Deriving structural rules for the parallel operator The structural rules can instead be obtained by analyzing the interface J_D of the minimal transition, whose nodes represent the "handles" of D for interacting with the environment. Since for each minimal transitiony J_D always contains the root nodes of D, then we can add a graph I representing a process Q in parallel with D, by obtaining a graph $J \rightarrowtail G$, where the interface J consists of J_D together with the free names of Q. Now, since there exist a mono $D \rightarrowtail G$ (because G consists of the graph D in parallel with I) and a morphism $J_D \rightarrow J$, such that the square (2) in Figure 4.14 commutes, the graph $J \rightarrowtail G$ can perform a BC rewriting step in $\mathcal{R}_{amb}$ with label $J \rightarrowtail F \leftarrowtail K$, where F and K consist of F_D and K_D together with the free names of Q. Process-wise, this means that if $P \xrightarrow{C[-]} P\epsilon$, then for the process $P|Q$ there is also a transition labelled $C[-]$.

Let us consider again the minimal transition for D_{in_1}. By analyzing the interface $J_{D_{in_1}}$ we may obtain the structural rule for the parallel operator. Since $J_{D_{in_1}}$ contains the root nodes of the graph D, then we can add a graph I representing a process Q in parallel with D. In this way, we obtain a graph $J \rightarrowtail G$, where the interface J consists of $J_{D_{in_1}}$ together with the free names of Q. The graph $J \rightarrowtail G$ can perform a BC rewriting step in $\mathcal{R}_{amb}$ with label $J \rightarrowtail F \leftarrowtail K$, where F and K respectively consist of $F_{D_{in_1}}$ and $K_{D_{in_1}}$ together with the free names of Q. This means that the graph context $J \rightarrowtail F \leftarrowtail K$ also represents the process context $x[-|X_1]|m[X_2]$. In terms of processes, this means that if $P \xrightarrow{x[-|X_1]|m[X_2]} P\epsilon$, then $P|Q$ also has a transition labeled with $x[-|X_1]|m[X_2]$. The resulting process is represented by the graph H which is obtained simply by replacing D_{in_1} with R_{in} in G. T,he node $\bullet_{2_p}$ after the reduction execution is under the ambient x and moreover it represents a process variable in P_ϵ. This means that the graph

(InTau)
$$\frac{}{n[in\ m.P|Q]|m[R]\xrightarrow{-}m[n[P|Q]|R]}$$

(OutTau)
$$\frac{}{m[n[out\ m.P|Q]|R]\xrightarrow{-}n[P|Q]|m[R]}$$

(OpenTau)
$$\frac{}{open\ n.P|n[Q]\xrightarrow{-}P|Q}$$

(TauAmb)
$$\frac{P\xrightarrow{-}P'}{n[P]\xrightarrow{-}n[P']}$$

(TauPar)
$$\frac{P\xrightarrow{-}P'}{P|Q\xrightarrow{-}P'|Q}$$

(TauRes)
$$\frac{P\xrightarrow{-}P'}{(\nu a)P\xrightarrow{-}(\nu a)P'}$$

(In)
$$\frac{}{in\ m.P_1\xrightarrow{x[-|X_1]|m[X_2]}m[x[P_1|X_1]|X_2]}$$

(InPar)
$$\frac{P\xrightarrow{x[-|X_1]|m[X_2]}P_\epsilon}{P|Q\xrightarrow{x[-|X_1]|m[X_2]}P_\epsilon\{{}^{Q|X_1}/_{X_1}\}}$$

(InRes)
$$\frac{P\xrightarrow{x[-|X_1]|m[X_2]}P_\epsilon \quad a\neq m}{(\nu a)P\xrightarrow{x[-|X_1]|m[X_2]}(\nu a)P_\epsilon}$$

(InAmb)
$$\frac{P\xrightarrow{x[-|X_1]|m[X_2]}P_\epsilon}{n[P]\xrightarrow{-|m[X_2]}P_\epsilon\{{}^{n}/_{x},{}^{\mathbf{0}}/_{X_1}\}}$$

(InAmbPar)
$$\frac{P\xrightarrow{-|m[X_2]}P_\epsilon}{P|Q\xrightarrow{-|m[X_2]}P_\epsilon|Q}$$

(InAmbRes)
$$\frac{P\xrightarrow{-|m[X_2]}P_\epsilon \quad a\neq m}{(\nu a)P\xrightarrow{-|m[X_2]}(\nu a)P_\epsilon}$$

(CoIn)
$$\frac{}{m[P_1]\xrightarrow{-|x[in\ m.X_1|X_2]}m[x[X_1|X_2]|P_1]}$$

(CoInPar)
$$\frac{P\xrightarrow{-|x[in\ m.X_1|X_2]}P_\epsilon}{P|Q\xrightarrow{-|x[in\ m.X_1|X_2]}P_\epsilon|Q}$$

(CoInRes)
$$\frac{P\xrightarrow{-|x[in\ m.X_1|X_2]}P_\epsilon \quad a\neq m}{(\nu a)P\xrightarrow{-|x[in\ m.X_1|X_2]}(\nu a)P_\epsilon}$$

(Out)
$$\frac{}{out\ m.P_1\xrightarrow{m[x[-|X_1]|X_2]}m[X_2]|x[P_1|X_1]}$$

(OutPar)
$$\frac{P\xrightarrow{m[x[-|X_1]|X_2]}P_\epsilon}{P|Q\xrightarrow{m[x[-|X_1]|X_2]}P_\epsilon\{{}^{Q|X_1}/_{X_1}\}}$$

(OutRes)
$$\frac{P\xrightarrow{m[x[-|X_1]|X_2]}P_\epsilon \quad a\neq m}{(\nu a)P\xrightarrow{m[x[-|X_1]|X_2]}(\nu a)P_\epsilon}$$

(OutAmb)
$$\frac{P\xrightarrow{m[x[-|X_1]|X_2]}P_\epsilon}{n[P]\xrightarrow{m[-|X_2]}P_\epsilon\{{}^{n}/_{x},{}^{\mathbf{0}}/_{X_1}\}}$$

(OutAmbPar)
$$\frac{P\xrightarrow{m[-|X_2]}P_\epsilon}{P|Q\xrightarrow{m[-|X_2]}P_\epsilon\{{}^{Q|X_2}/_{X_2}\}}$$

(OUtAmbRes)
$$\frac{P\xrightarrow{m[-|X_2]}P_\epsilon \quad a\neq m}{(\nu a)P\xrightarrow{m[-|X_2]}(\nu a)P_\epsilon}$$

(Open)
$$\frac{}{open\ n.P_1\xrightarrow{-|n[X_1]}P_1|X_1}$$

(OpenPar)
$$\frac{P\xrightarrow{-|n[X_1]}P_\epsilon}{P|Q\xrightarrow{-|n[X_1]}P_\epsilon|Q}$$

(OpenRes)
$$\frac{P\xrightarrow{-|n[X_1]}P_\epsilon \quad a\neq n}{(\nu a)P\xrightarrow{-|n[X_1]}(\nu a)P_\epsilon}$$

(CoOpen)
$$\frac{}{n[P_1]\xrightarrow{-|open\ n.X_1}P_1|X_1}$$

(CoOpenPar)
$$\frac{P\xrightarrow{-|open\ n.X_1}P_\epsilon}{P|Q\xrightarrow{-|open\ n.X_1}P_\epsilon|Q}$$

(CoOpenRes)
$$\frac{P\xrightarrow{-|open\ n.X_1}P_\epsilon \quad a\neq n}{(\nu a)P\xrightarrow{-|open\ n.X_1}(\nu a)P_\epsilon}$$

Figure 4.19: The LTS $\mathcal{S}$.

modeling Q (that has as root the node $\bullet_{2_p}$) after the reduction execution is under the ambient x and it is in parallel with the process variable X_1. Therefore the resulting process is $P_\epsilon\{^{Q|X_1}/_{X_1}\}$.

Deriving structural rules for the restriction operator Also the structural rules for the restriction operator can be obtained by analyzing the interface J_D. Indeed, we know that a graph $J \rightarrowtail G$ representing a process P can perform a BC rewriting step if and only if there exist a mono $D \rightarrowtail G$ and a morphism $J_D \to J$, such that the square (2) in Figure 4.14 commutes. If we modify the interface J by removing one or more name nodes, then the graph G with the new interface J' can also perform the same BC rewriting step if and only if there exists a morphism $J_D \to J'$, such the square (2) in Figure 4.14 commutes. This means that all the name nodes of J_D must also belong to J', therefore as suggested by the encoding, the ambient names of P that are in J_D cannot be restricted. In terms of processes, this means that if $P \xrightarrow{C[-]} P\epsilon$, then for the process $(\nu a)P$ there is also a transition labelled $C[-]$ if the names belonging to J_D do not belong to a.

On the basis of the remarks above, starting from the minimal transition for D_{in_1}, we can derive a structural rule for the restriction operator. In particular, if $P \xrightarrow{x[-|X_1]|m[X_2]} P\epsilon$, then for the process $(\nu a)P$ there is a transition with the same label leading to the process $(\nu a)P_\epsilon$ if the name m (that belongs to the interface $J_{D_{in_1}}$) is not restricted.

Note that the interface J_D also allows us to obtain a graph $J \rightarrowtail G$ that is composed of the graph D with another graph on the top. It is easy to note that in this case the graph $J \rightarrowtail G$ does not perform any BC transition because it is impossible to find a morphism $J_D \to J$ such the square (2) in Figure 4.14 commutes.

Equivalence between LTSs As for $\mathcal{D}$, also for $\mathcal{S}$ we define the LTS $\mathcal{S}_\mathcal{I}$ for pure processes by instantiating all the variables of the labels and of the resulting states. Moreover, also in this case, we implicitly assume that it is closed with respect to the structural congruence.

Definition 4.7. *Let P, Q be pure processes and let $C[-]$ be a pure context. Then, we have that $P \xrightarrow{C[-]}_{\mathcal{S}_\mathcal{I}} Q$ if there exists a transition $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ and a substitution σ such that $Q_\epsilon\sigma = Q$ and $C_\epsilon[-]\sigma = C[-]$.*

As stated by the following theorem, the LTSs $\mathcal{S}_\mathcal{I}$ and $\mathcal{D}_\mathcal{I}$ coincide.

Theorem 4.2. *Let P be a pure process and let $C[-]$ be a pure context. Then, $P \xrightarrow{C[-]}_{\mathcal{D}_\mathcal{I}} Q$ if and only if $P \xrightarrow{C[-]}_{\mathcal{S}_\mathcal{I}} Q$.*

The proof of Theorem 4.2 is shown in Section B.1 (Appendix B).

4.8 Equivalence between LTSs

This section shows the equivalence between our LTS $\mathcal{S}_\mathcal{I}$ defined on pure processes and the LTS proposed by Rathke and Sobociński in [60, Figures 6, 7 and 8].

Their LTS is organized into three components: the process-view LTS $\mathcal{C}$, the context-view LTS $\mathcal{A}$, and the combined LTS $\mathcal{CA}$. The labels of the LTS $\mathcal{CA}$ have the shape $\alpha \downarrow \vec{M}$, where α is derived by the LTS $\mathcal{C}$, and $\vec{M}$ by the LTS $\mathcal{A}$. In a transition $P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} Q$, the label α identifies the minimal context needed by the pure process P to react, while $\vec{M}$ is a list of pure processes and ambient names, representing an instantiation of the context components. The first column of Table 4.1 shows all the labels α of the LTS $\mathcal{C}$, while the second column for each of these labels presents the context χ_α that it identifies. We refer the interested reader to [60, Lemma 6] for a more detailed explanation of this correspondence. Note that each context χ_α contains a set of typed numbered holes. In particular, holes of type N can be instantiated with ambient names, while holes of type Pr can be instantiated with pure processes. Therefore, in a transition $P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} P'$, the tuple $\vec{M}$ has to provide an instantiation for all context components, that is, for each hole of χ_α of process type Pr (different from 1_{Pr}) and for each hole of name type N. The hole 1_{Pr} instead represents the hole that has to be instantiated with the process P. As we are going to see later on, it is the hole that is represented with $-$ in our contexts. For example, consider the transition $P \xrightarrow{in\, m\downarrow\vec{M}}_{\mathcal{CA}} P'$. Here the tuple $\vec{M}$ must provide an instantiation for the holes 2_{Pr}, 3_N and 4_{Pr} of the context $\chi_{in\, m}$. This means that it has to have the following shape $\vec{M} : Q, n, R$, for Q, R pure processes and n ambient name.

It is immediate to note that there exists a one-to-one correspondence between the labels $C_\epsilon[-]$ of our LTS $\mathcal{S}$ and the contexts χ_α listed in the second column of Table 4.1. This correspondence is shown in the same table, where $C^\alpha_\epsilon[-]$ (the third column) denotes the label of our LTS $\mathcal{S}$ corresponding to the context χ_α of the second column.

Note that for each label α, the contexts $C^\alpha_\epsilon[-]$ and χ_α have the same shape. The hole $-$ in $C^\alpha_\epsilon[-]$ corresponds to the hole 1_{Pr} in χ_α, and there is a correspondence between name and process variables of $C^\alpha_\epsilon[-]$ and holes of χ_α of type N an Pr, respectively. Consider e.g. the label $C^{in\,m}_\epsilon[-] = x[-|X_1]|m[X_2]$ and the corresponding context $\chi_{in\,m} = 3_N[1_{Pr}|2_{Pr}]|m[4_{Pr}]$. The two contexts have the same shape. In particular, the name variable x corresponds to the hole 3_N, the hole $-$ corresponds to the hole 1_{Pr} , and the process variables X_1 and X_2 correspond to the holes 2_{Pr} and 4_{Pr}, respectively.

As explained in Section 4.7, a substitution σ for a context $C_\epsilon[-]$ provides an instantiation for the process and name variables of the context. For instance, a substitution σ for the context $C^{in\,m}_\epsilon[-] = x[-|X_1]|m[X_2]$ must have the shape $\{^Q/_{X_1}, ^n/_x, ^R/_{X_2}\}$ for Q, R pure processes and n ambient name. Now, since there is a correspondence between holes of a context χ_α of type N and Pr, and name and process variables of the relative context $C^\alpha_\epsilon[-]$, respectively, it is obvious that given a tuple $\vec{M}$ for χ_α it is possible to determine a unique corresponding substitution σ_M for $C^\alpha_\epsilon[-]$. Such a substitution σ_M instantiates each variable with the same value used by $\vec{M}$ to instantiate the hole corresponding to that variable. Analogously, given a substitution σ for $C^\alpha_\epsilon[-]$, it is possible to determine a unique corresponding substitution $\vec{M}_\sigma$ for χ_α. Consider again the context $\chi_{in\,m}$ and the tuple $\vec{M} = Q, n, R$ providing an instantiation respectively for the holes 2_{Pr}, 3_N and 4_{Pr}. The substitution σ_M (induced by $\vec{M}$) for the corresponding context $C^{in\,m}_\epsilon[-] = x[-|X_1]|m[X_2]$ is $\{^Q/_{X_1}, ^n/_x, ^R/_{X_2}\}$. Analogously, it is possible to determine the tuple $\vec{M}$ from the substitution σ_M. The last two columns of Table 4.1 show for each α respectively the shape of the tuples $\vec{M}^\alpha_\sigma$ and σ^α_M.

The following propositions allow us to formally prove the correspondence between the LTS $\mathcal{S}_I$ and the LTS $\mathcal{CA}$. Their proofs are in Section B.2 (Appendix B).

Proposition 4.3. Let P be a pure process. If $P \xrightarrow{\alpha\downarrow\vec{M}^\alpha}_{\mathcal{CA}} Q$, then there exists Q_ϵ such that $P \xrightarrow{C^\alpha_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ and $Q \equiv Q_\epsilon\sigma^\alpha_M$.

Proposition 4.4. Let P be a pure process and let σ be a substitution. If $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ and $Q_\epsilon\sigma \equiv Q$, then there exists α such that $C_\epsilon[-] = C^\alpha_\epsilon[-]$ and $P \xrightarrow{\alpha\downarrow\vec{M}^\alpha_\sigma}_{\mathcal{CA}} Q$.

From the two propositions above and from the definition of the LTS $\mathcal{S}_\mathcal{I}$ (Definition 4.7) follows the main result of this section.

Theorem 4.3. *Let P be a pure process. If $P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} Q$, then there is a unique (up-to $\equiv$) substitution σ such that $P \xrightarrow{C^\alpha_\epsilon[-]\sigma}_{\mathcal{S}_\mathcal{I}} Q$. Vice versa, if $P \xrightarrow{C[-]}_{\mathcal{S}_\mathcal{I}} Q$, then there are α and a unique (up-to $\equiv$) tuple $\vec{M}$ such that $C[-] = C^\alpha[-]$ and $P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} Q$.*

α	χ_α	$C^\alpha_\epsilon[-]$	$\vec{M}^\alpha_\sigma$	σ^α_M
$in\ m$	$3_N[1_{Pr}\mid 2_{Pr}]\mid m[4_{Pr}]$	$x[-\mid X_1]\mid m[X_2]$	PnQ	$\{^P/_{X_1}, ^n/_x, ^Q/_{X_2}\}$
$[in\ m]$	$1_{Pr}\mid m[2_{Pr}]$	$-\mid m[X_2]$	P	$\{^P/_{X_2}\}$
$[\overline{in\ m}]$	$4_N[in\,m.2_{Pr}\mid 3_{Pr}]\mid 1_{Pr}$	$-\mid x[in\ m.X_1\mid X_2]$	PQn	$\{^P/_{X_1}, ^Q/_{X_2}, ^n/_x\}$
$out\ m$	$m[3_N[1_{Pr}\mid 2_{Pr}]\mid 4_{Pr}]$	$m[x[-\mid X_1]\mid X_2]$	PnQ	$\{^P/_{X_1}, ^n/_x, ^Q/_{X_2}\}$
$[out\ m]$	$m[1_{Pr}\mid 2_{Pr}]$	$m[-\mid X_2]$	P	$\{^P/_{X_2}\}$
$open\ n$	$1_{Pr}\mid n[2_{Pr}]$	$-\mid n[X_1]$	P	$\{^P/_{X_1}\}$
$\overline{open\ n}$	$open\,n.2_{Pr}\mid 1_{Pr}$	$-\mid open\ n.X_1$	P	$\{^P/_{X_1}\}$
τ	1_{Pr}	$-$	$\emptyset$	$\{\}$

Table 4.1: The correspondence between α, χ_α, $C^\alpha_\epsilon[-]$, $\vec{M}^\alpha_\sigma$ and σ^α_M.

4.9 A Labelled Transition System for the Asynchronous CCS

In this section we present an IPO-LTS on processes of the asynchronous CCS. To this end, as for the mobile ambients calculus, we shortly introduce an extended version of the asynchronous CCS, where processes containing process and summation variables are allowed.

Extended Asynchronous CCS. Figure 4.20 shows the extended syntax of the calculus. We assume a set $\mathcal{N}$ of *names* ranged over by $a, b, c, \ldots$. As for mobile ambients, we include a set of *process variables* $\mathcal{X} = \{X, Y, \ldots\}$, which are needed for the presentation of the LTS in Figure 4.21, and a set of *summation variables* $\mathcal{M} = \{X_M, Y_M, \ldots\}$, useful to discuss the application of the BC mechanism to graphs encoding ACCS processes.

By following Definition 4.1 for mobile ambients, we define a *pure process* and a pure summation as an extended process, respectively summation, such that no process or summation variable occurs. We let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of pure processes, and $M, N, \ldots$ over the set $\mathcal{S}$ of *pure summation*. We use the standard definitions for the set of free names of a pure process P, denoted by $fn(P)$, and for α-convertibility, with respect to the restriction operators (νn). As for the general definition, variables carry no name, hence $fn(X) = \emptyset$ and $fn(X_M) = \emptyset$.

As for the mobile ambients calculus, we consider a family of *substitutions*, which may replace a process/summation variable with a pure process/summation. Substitutions avoid name capture: for a pure process P, the expression $(\nu a)(\nu b)(a.b.X + X_M)\{^{\overline{a}}/_X, ^{b.P}/_{X_M}\}$ corresponds to the pure process $(\nu c)(\nu d)(c.d.\overline{a} + b.P)$, for $c, d \notin \{b\} \cup fn(b.P)$.

The semantics of the calculus is given by the reduction relation and the structural congruence on pure processes both defined in Figure 3.28.

Borrowed Contexts LTSs for asynchronous CCS. The graphical encoding for the asynchronous CCS presented in Section 3.9 is amenable to the BC mechanism. To this end, as for mobile ambients, we need to consider extended processes and, in order to apply the borrowed context technique, we need to have graphs with only one interface. We do not formally introduce the encoding for extended processes of the asynchronous CCS. It can indeed be obtained by following what we did for the mobile ambients calculus in Section 4.3. Intuitively, the encoding is defined as discussed in Section 3.9, with the only difference that here nodes representing free names of the process are in the input interface together with the $\bullet$ root node, the $\bullet$ nodes representing the process variables and the $\diamond$ nodes representing the summation variables.

The graph transformation system modeling the reduction semantics of the extended asynchronous CCS is exactly the same as the one presented in Section 3.9.

So, the BC synthesis mechanism may be applied to it in order to derive an LTS for graphs representing asynchronous CCS processes. Figure 4.27, 4.28 and 4.29 show three examples of BC transitions. In particular, the first one shows an application of the BC synthesis mechanism to the graphical encoding of the process $\tau.\mathbf{0}$, induced by the rule $p_\tau : L_\tau \leftarrowtail I_\tau \rightarrow R_\tau$. We take as D the left-hand side L_τ, therefore the starting graph needs no context for the reduction and so the label of this transition is the identity context. Intuitively, this corresponds to an internal transition over processes, labelled with τ.

Figure 4.28 shows an application of the BC synthesis mechanism to the graphical encoding of the process $a.\overline{a} + \tau.\mathbf{0}$, induced by the rule $p_{com} : L_{com} \leftarrowtail I_{com} \rightarrow R_{com}$. We take as D the subgraph of L_{com} representing an input prefix. The graph G^+ is the graph G in parallel with the graph representing an output over a, thus intuitively it represents the process $a.\overline{a} + \tau.\mathbf{0} \mid \overline{a}$. The graph $J \rightarrowtail G$, in order to reach the graph G^+, has to borrow from the environment the context that represents the process context $- \mid \overline{a}$. The graphs C and H are then constructed as in the standard DPO approach. Intuitively, $K \rightarrow H$ represents the process $\overline{a} \mid \mathbf{0}$. Summarizing, this transition models a communication over the channel a, where the output action is provided by the environment.

Finally, Figure 4.29 represents a BC derivation again induced by the rule p_{com}, but with starting graph the encoding of the process $\overline{a}$. Here we take as D the subgraph of L_{com} representing an output prefix. The

$$P ::= \overline{a},\ P_1 \mid P_2,\ (\nu a)P,\ M,\ X \qquad\qquad M ::= \mathbf{0},\ \alpha.P,\ M_1 + M_2,\ X_M \qquad\qquad \alpha ::= a,\ \tau$$

Figure 4.20: Extended syntax of ACCS.

graph G^+ is the graph G in parallel with the graph representing an input over a, thus intuitively it represents the process $\overline{a} \mid a.X_1 + X_M$, for some process variable X_1 and summation variable X_M. The graph $J \rightarrowtail G$, in order to reach the graph G^+, has to borrow from the environment the context $J \rightarrowtail F \leftarrowtail K$ representing the syntactic context $- \mid a.X_1 + X_M$. Note indeed that in the resulting interface K there are a process node $\bullet^{p_1}$ and a summation node $\diamond^{s_1}$ pointing, respectively, to the process node (modelling a process variable) of F following the input operator, and to the summation node (representing a summation variable) in F representing the root of the input operator. The graphs $K \rightarrow H$, intuitively, represents the $\mathbf{0}$ process, so summarizing, this transition models a communication over the channel a, where the input action is provided by the environment.

An LTS for asynchronous CCS processes. Here we do not present all the steps necessary to obtain the LTS directly defined over ACCS processes. It suffices to know that we mimicked [7], where the authors derived an LTS for the ordinary CCS by employing the borrowed context mechanism.

Figure 4.21 shows the LTS A.

Obviously the labels are minimal contexts, i.e., they represent the exact amount of context needed by a process to react. Moreover, note that the label of the (SND) rule contains the process variable X_1. Actually, it should also contain the summation variable, but, as it is possible to note in the BC transition shown in Figure 4.29, this variable does not occur in the arriving state, and it also plays no role in the derived bisimilarity. We therefore avoided considering it in the label.

Following Definition 4.6 for mobile ambients, we define the LTS A_I for processes over the not-extended syntax by instantiating the process variable of the labels and of the resulting states.

Rule RCV represents the main difference between the LTS A and the one derived in [7] for the synchronous version. Since in the asynchronous CCS outputs have no continuations, then the label and the target state have no process variable which is instead needed in the synchronous version of the calculus.

It is easy to see that there is a close correspondence between the ordinary LTS semantics (in Figure 3.29) and the LTS A: $P \xrightarrow{\tau} Q$ iff $P \xrightarrow{-} Q$, $P \xrightarrow{a} Q$ iff $P \xrightarrow{-|\bar{a}} Q$ and $P \xrightarrow{\bar{a}} Q$ iff $P \xrightarrow{-|a.X_1} Q|X_1$.

However, as we will see later, for the asynchronous CCS IPO-bisimilarity is too fine grained. Consider for example the two processes $a.\bar{a}+\tau.\mathbf{0}$ and $\tau.\mathbf{0}$. They are asynchronously bisimilar (according to Definition 3.19), but they are not IPO-bisimilar. In the next chapter we will introduce a new semantics for reactive systems that generalizes $\sim^A$.

4.10 Summary

In this chapter we exploit the graphical encodings respectively for mobile ambients and ACCS, both proposed in the previous chapter, to distill two LTSs on (processes encoded as) graphs. Each LTS is obtained semi-automatically by first applying the BC technique to the graph transformation system associated to the calculus, and further using some pruning techniques for removing possible reductions, yet preserving bisimilarity. The LTS defined on graphs is then exploited in order infer an LTS directly defined on processes. In particular, as far as ACCS, we do not present in detail the procedure needed to obtain the LTS on process. This because it is very similar to the one used in [7] for its synchronous version. For mobile ambients, also a suitable set of SOS rules for the calculus is presented, showing that the LTS $\mathcal{S}$ they induce coincides with the derived one. Finally, exploiting the SOS presentation, we prove that our $\mathcal{S}$ is actually equivalent with an alternative proposal presented in [60].

In spite of the great interest received by mobile ambients, there are relatively few works concerning a labelled characterization of the calculus. After early attempts by Cardelli and Gordon [37] and (*via*

$$\text{(TAU)}\ \frac{P \rightarrow Q}{P \xrightarrow{-} Q} \qquad \text{(RCV)}\ \frac{P \equiv (\nu A)(a.Q + M|R) \quad a \notin A}{P \xrightarrow{-|\bar{a}} (\nu A)(Q|R)} \qquad \text{(SND)}\ \frac{P \equiv (\nu A)(\bar{a}|Q) \quad a \notin A}{P \xrightarrow{-|a.X_1} (\nu A)(Q|X_1)}$$

Figure 4.21: The LTS A

a graphical encoding) by Ferrari, Montanari and Tuosto [30], the only papers that we are aware of are by Merro and Zappa-Nardelli [47] and by Rathke and Sobociński [60]. We already addressed the LTS introduced in the latter: we only remark that, analogously to our work, Rathke and Sobociński employ a general systematic procedure for deriving LTSs that they previously introduced in [59]. As for the former, the LTS proposed by Merro and Zappa-Nardelli is restricted to *systems*, i.e., those processes obtained by the parallel composition of ambients. For this reason, our rules IN, OPEN and OUT have no counterpart in [47]. Instead, the rules INAMB, COIN and OUTAMB exactly correspond to the rules (Enter), (Co-Enter), (Exit) in Table 6 of [47]. Moreover, our rule COOPEN roughly corresponds to their (Open). Indeed the former inserts a process into the context $-|open\ n.X_1$, while the latter into $k[-|open\ n.X_1|X_2]$ (again, this difference is due to the fact that the LTS of [47] is restricted to systems). It is important to note that, differently from our LTS, the labels of the rules (Enter) and (Exit) contain the name of the migrating ambient n. This requires defining two extra rules (Enter Shh) and (Exit Shh) for the case when n is restricted.

For a practitioner, the main interest of the results presented in this chapter lies on the presentation of a succinct LTS for mobile ambients, and the associated set of SOS rules. However, we do believe that our work represents a relevant case study for the theory of reactive systems [45]. As already pointed out in the introduction, BC rewriting and bigraphical reactive systems [51] are both instances of this theory. Our work, together with [7], shows that the borrowed contexts approach is quite effective in deriving LTS for process calculi. In particular, it seems to confirm the advantage of borrowed contexts over graphs with interfaces with respect to bigraphs. In bigraphs, all the reduction rules must be ground (i.e., they can not contain process variables). As a result, also the labels and the arriving states of the derived transitions must be ground. Instead, rewriting with BCs allows to employ few non ground rules and thus the resulting transitions have labels and arriving states containing (process and name) variables. This feature was not relevant for calculi such as asynchronous CCS, its synchronous version and π, because the variables in the labels always occur "outside" of the arriving state and thus can be forgotten. As an example, consider the asynchronous CCS transition $b.\mathbf{0} \mid \bar{a} \xrightarrow{-|a.X} b.\mathbf{0} \mid X$ derived from the (non ground) reduction rule $\bar{a} \mid a.X \rightarrow X$. The behaviour of the process $b.\mathbf{0} \mid X$ is trivially equivalent to b: their interaction is basically restricted to processes offering a $\bar{b}$ action, and we can thus avoid to consider X. Instead, in the case of mobile ambients, the ability of considering non ground states is fundamental, because process variables may occur nested inside ambients in arriving states.

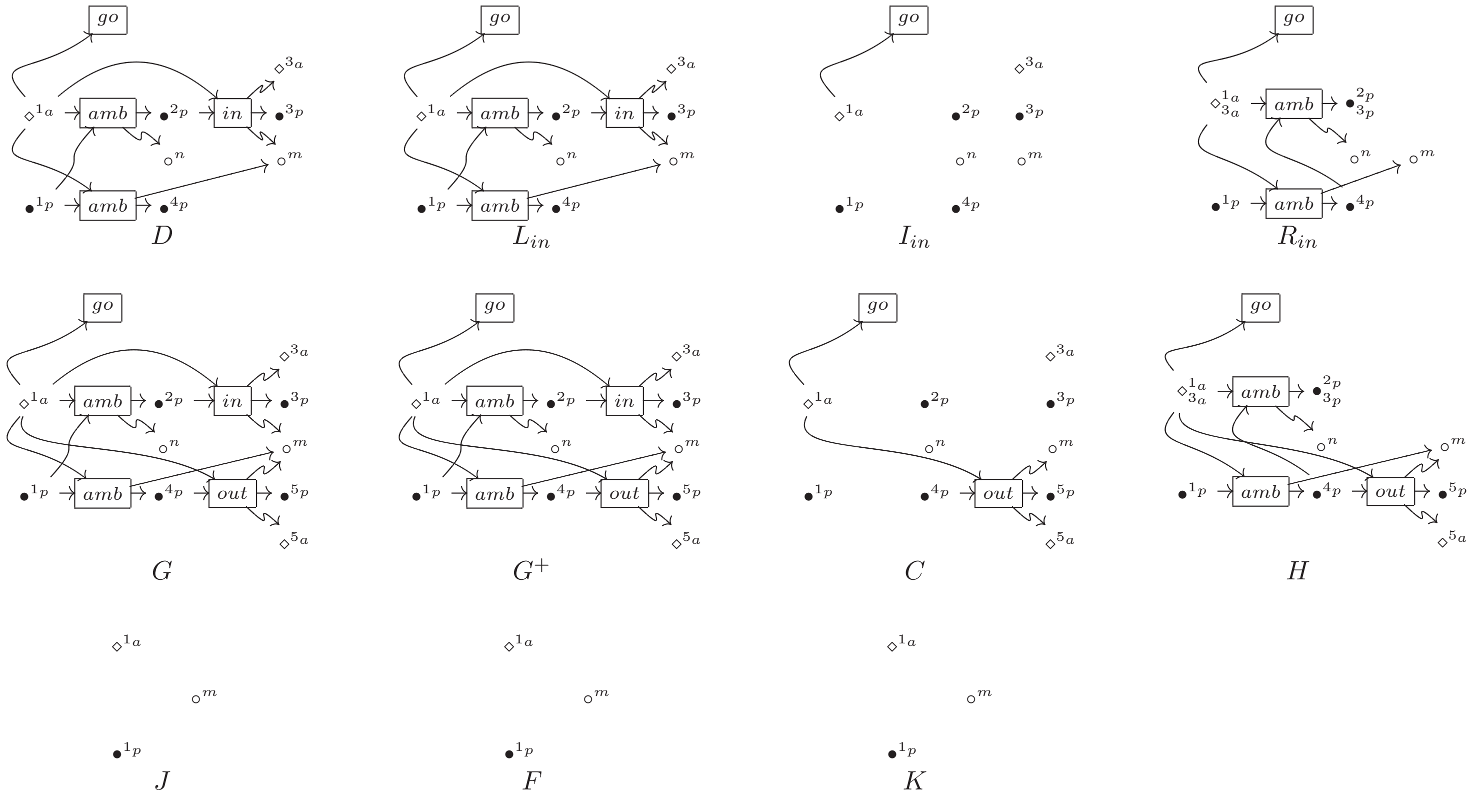


Figure 4.22: Ambient n enters ambient m. This corresponds to the transition $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]) \xrightarrow{-} (\nu n)(m[n[\mathbf{0}]|out\ m.\mathbf{0}])$.

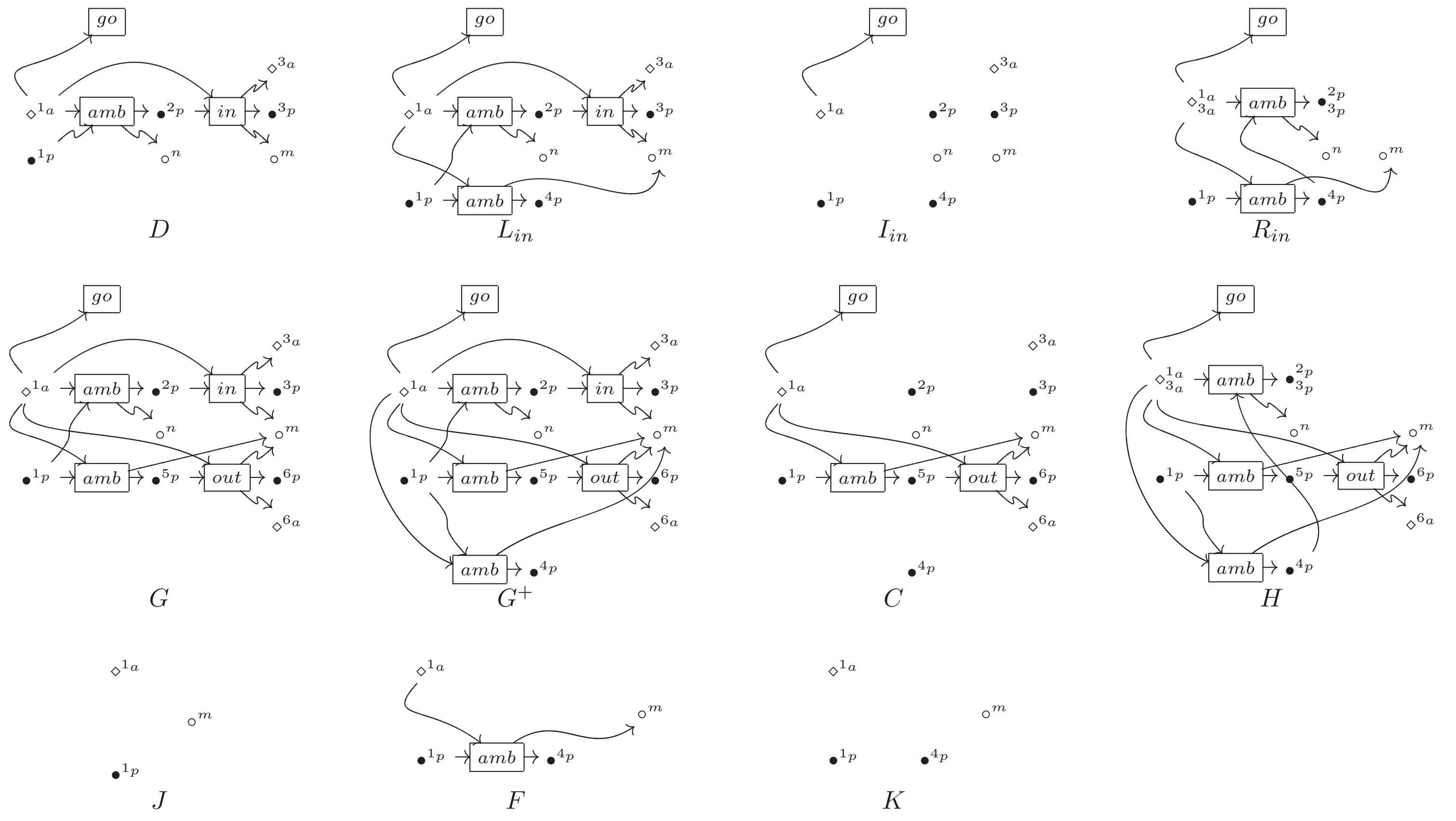


Figure 4.23: Ambient n enters ambient m (from environment). This corresponds to the transition $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]) \xrightarrow{-|m[X]} (\nu n)(m[out\ m.\mathbf{0}]|m[n[\mathbf{0}]|X])$.

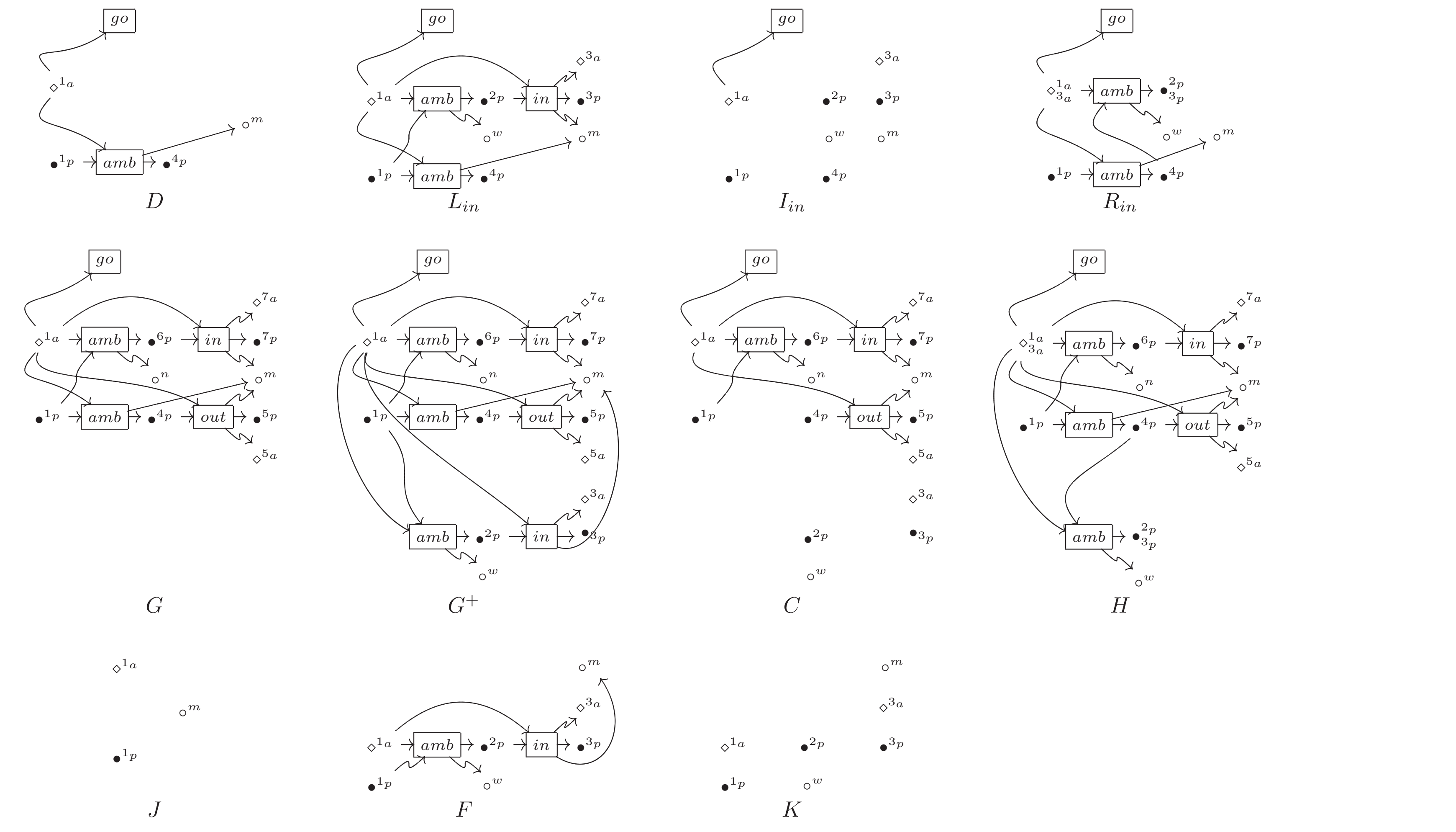


Figure 4.24: Ambient w (from environment) enters ambient m. This corresponds to the transition $(\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}]) \xrightarrow{-|w[in\ m.X_2|X_1]} (\nu n)(n[in\ m.\mathbf{0}]|m[out\ m.\mathbf{0}|w[X_2|X_1]])$.

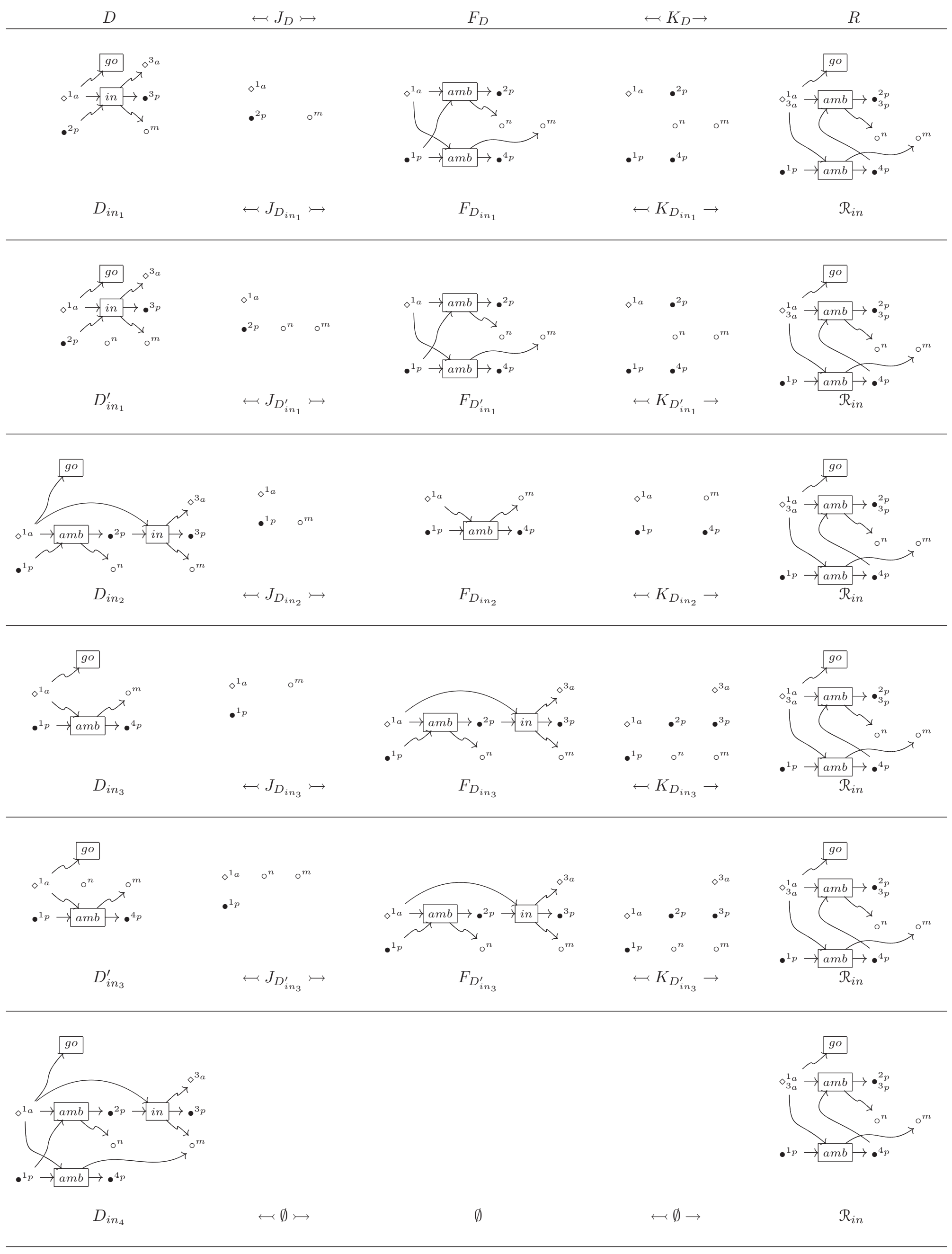

D
$\longleftarrow J_D \rightarrowtail$
F_D
$\longleftarrow K_D \rightarrow$
R
D_{in_1}
$\longleftarrow J_{D_{in_1}} \rightarrowtail$
$F_{D_{in_1}}$
$\longleftarrow K_{D_{in_1}} \rightarrow$
$\mathcal{R}_{in}$
D'_{in_1}
$\longleftarrow J_{D'_{in_1}} \rightarrowtail$
$F_{D'_{in_1}}$
$\longleftarrow K_{D'_{in_1}} \rightarrow$
$\mathcal{R}_{in}$
D_{in_2}
$\longleftarrow J_{D_{in_2}} \rightarrowtail$
$F_{D_{in_2}}$
$\longleftarrow K_{D_{in_2}} \rightarrow$
$\mathcal{R}_{in}$
D_{in_3}
$\longleftarrow J_{D_{in_3}} \rightarrowtail$
$F_{D_{in_3}}$
$\longleftarrow K_{D_{in_3}} \rightarrow$
$\mathcal{R}_{in}$
D'_{in_3}
$\longleftarrow J_{D'_{in_3}} \rightarrowtail$
$F_{D'_{in_3}}$
$\longleftarrow K_{D'_{in_3}} \rightarrow$
$\mathcal{R}_{in}$
D_{in_4}
$\longleftarrow \emptyset \rightarrowtail$
$\emptyset$
$\longleftarrow \emptyset \rightarrow$
$\mathcal{R}_{in}$
go
in
amb

Figure 4.25: The minimal transitions generated by the rule p_{in}.

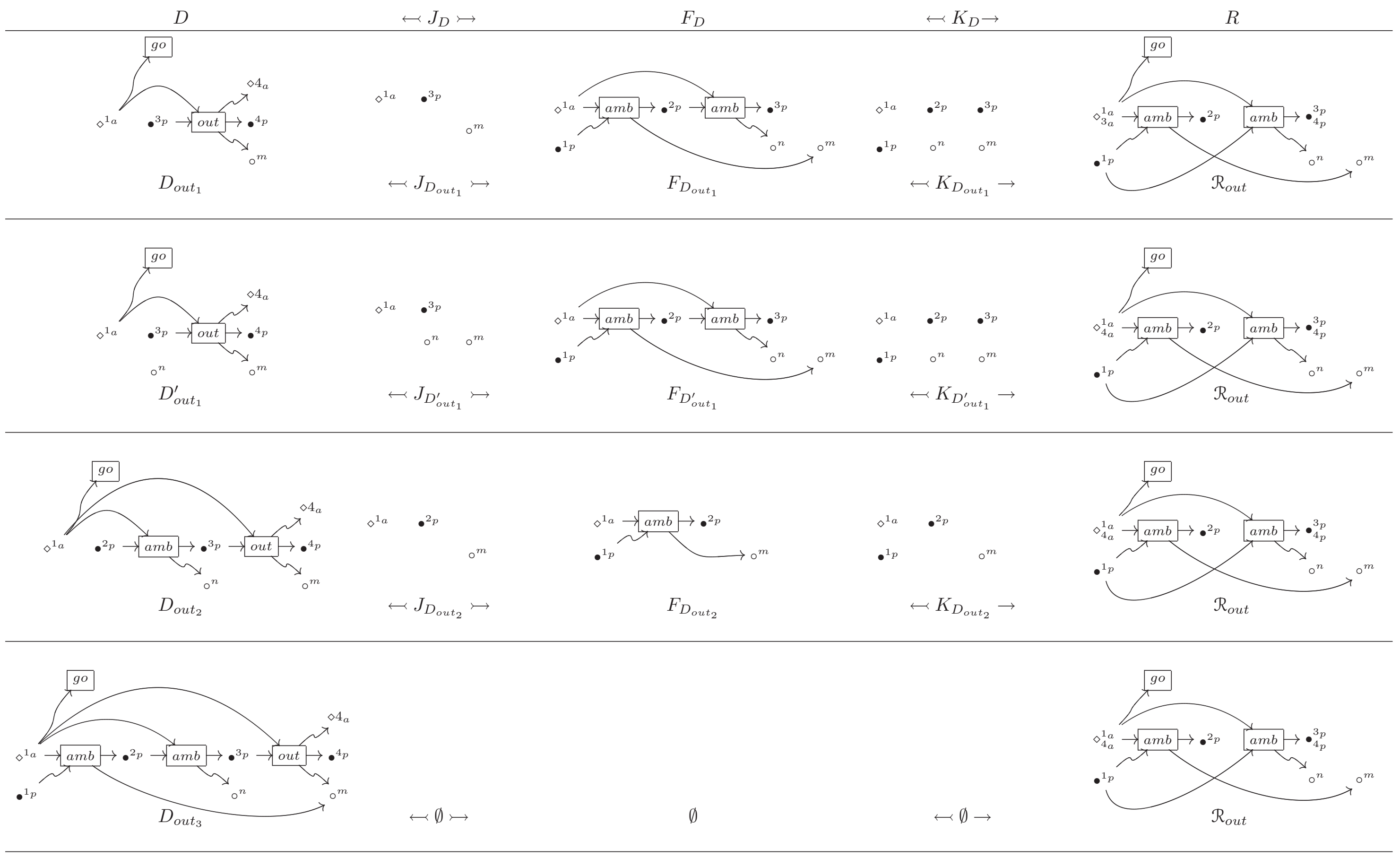

D
$\leftarrowtail J_D \rightarrowtail$
F_D
$\leftarrowtail K_D \rightarrow$
R
D_{out_1}
$\leftarrowtail J_{D_{out_1}} \rightarrowtail$
$F_{D_{out_1}}$
$\leftarrowtail K_{D_{out_1}} \rightarrow$
$\mathcal{R}_{out}$
D'_{out_1}
$\leftarrowtail J_{D'_{out_1}} \rightarrowtail$
$F_{D'_{out_1}}$
$\leftarrowtail K_{D'_{out_1}} \rightarrow$
$\mathcal{R}_{out}$
D_{out_2}
$\leftarrowtail J_{D_{out_2}} \rightarrowtail$
$F_{D_{out_2}}$
$\leftarrowtail K_{D_{out_2}} \rightarrow$
$\mathcal{R}_{out}$
D_{out_3}
$\leftarrowtail \emptyset \rightarrowtail$
$\emptyset$
$\leftarrowtail \emptyset \rightarrow$
$\mathcal{R}_{out}$
go
amb
out

Figure 4.26: The minimal transitions generated by the rule p_{out}.

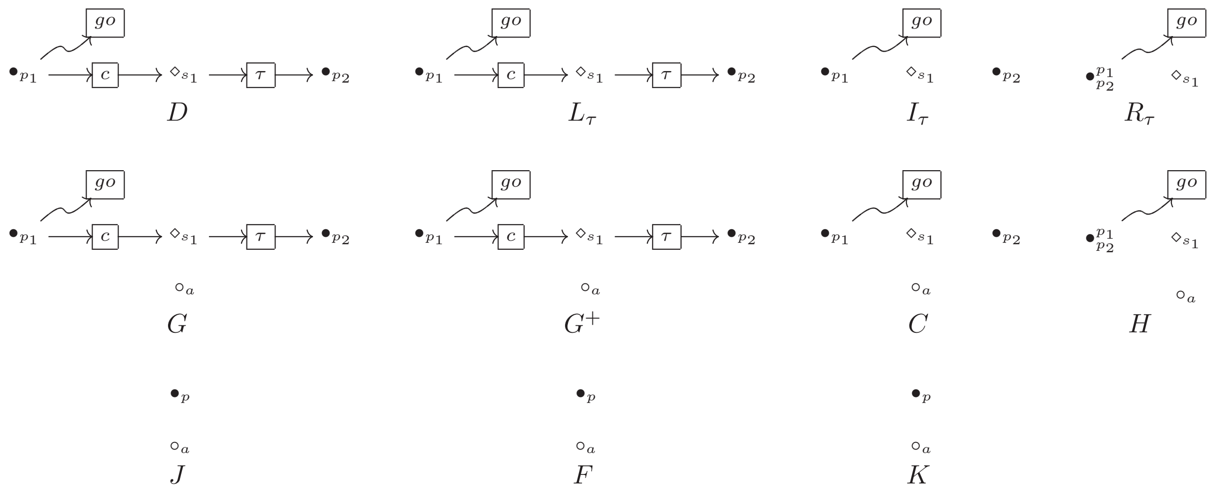


Figure 4.27: The BC transition corresponding to $\tau.\mathbf{0} \xrightarrow{-} \mathbf{0}$.

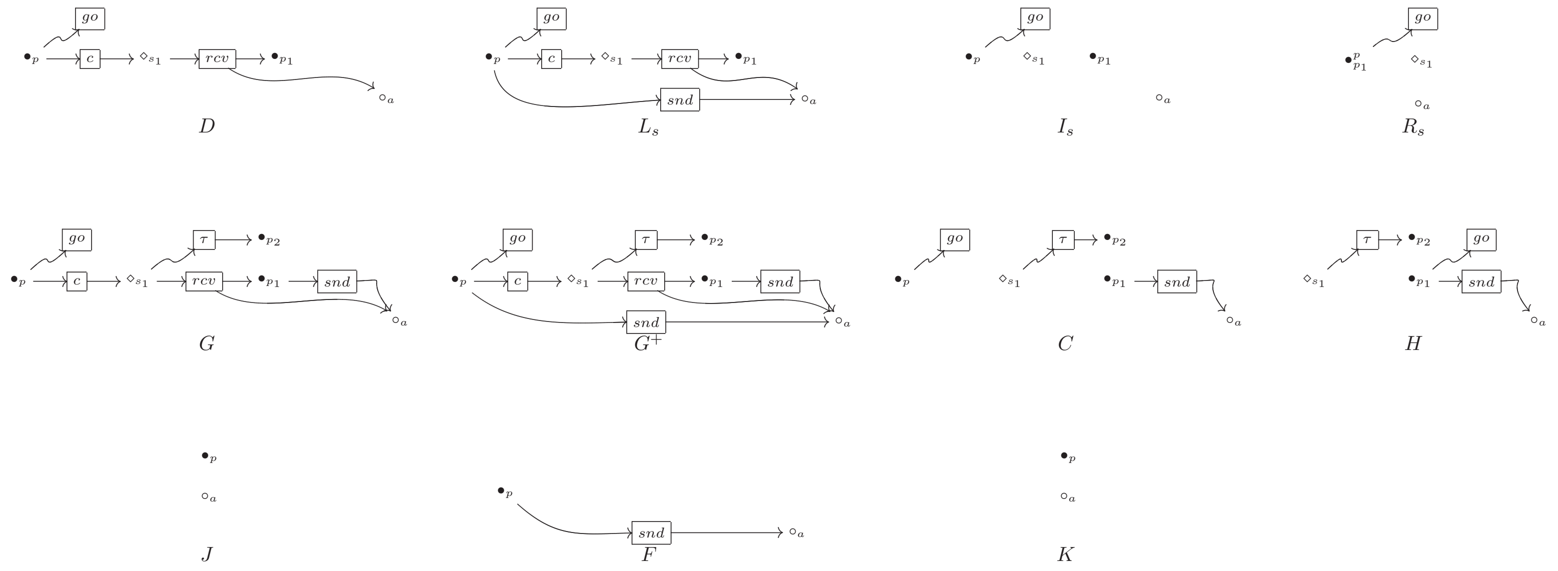


Figure 4.28: The BC transition corresponding to the transition $a.\overline{a} + \tau.\mathbf{0} \xrightarrow{-|\overline{a}} \overline{a}|\mathbf{0}$.

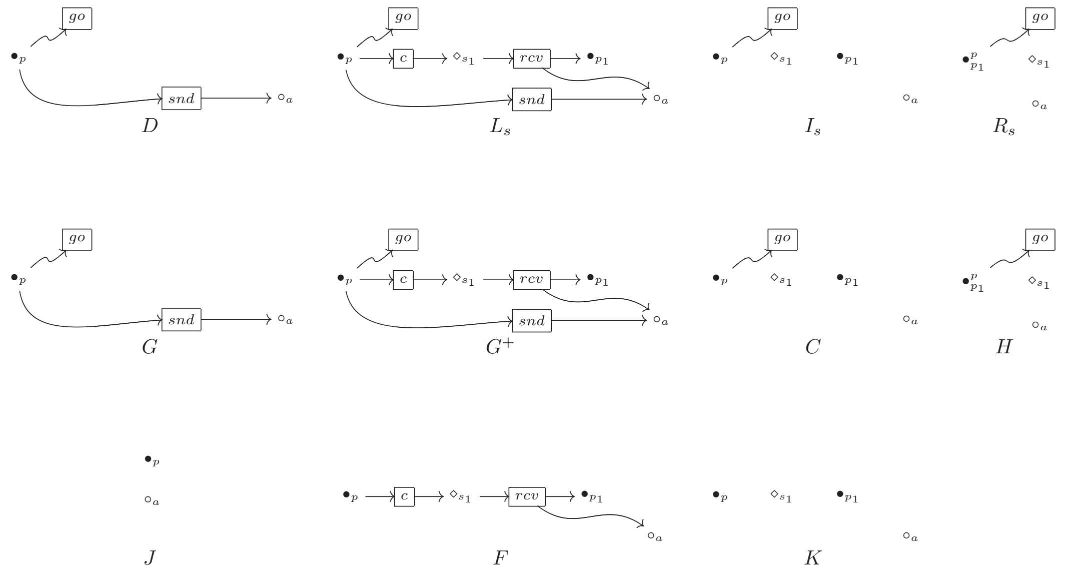


Figure 4.29: The BC transition corresponding to the transition $\overline{a} \xrightarrow{-|a.X_1+X_M} \mathbf{0}$.

Chapter 5

Barbed semantics for reactive systems

As said in Section 2.1, reactive systems represent a meta-framework aimed at deriving behavioral congruences for those specification formalisms whose operational semantics is provided by rewriting rules. Despite its applicability, they suffered so far from some drawbacks. Among them, one of the most important is that the efforts focused on strong bisimilarity, disregarding weak and barbed semantics. As far as the weak semantics is concerned, the only proposal we are aware of is in [43], where the author introduces a notion of weak bisimilarity for bigraphs.

In this chapter we address this issue, by providing suitable notions of barbed and weak barbed saturated semantics for reactive systems, and their characterization via transition systems labelled with minimal contexts, by exploiting the semi-saturated game, where a minimal context may be matched by any context.

The results above may have potentially far reaching consequences on the usability of the reactive systems formalism. However, their adequacy has to be properly established, by checking it against suitable case studies. To this end, we instantiate our proposal over the calculus of mobile ambients, whose observational semantics is still in a flux, and over the asynchronous CCS. In particular, for mobile ambients, we prove that our proposal captures the behavioural semantics for the calculus proposed by Rathke and Sobocinski and by Merro and Zappa Nardelli, while for the asynchronous CCS we show that it is able to capture the standard asynchronous bisimilarity (Definition 3.19).

The chapter is organized as follows. Section 5.1 discusses the motivations leading us to introduce a new semantics for reactive systems. Section 5.2 recalls the strong and weak behavioural equivalences for the mobile ambients calculus. Section 5.3 presents the technical core of the chapter, the introduction of barbed and weak barbed semantics for reactive systems, and offers a labelled characterization by means of their semi-saturated counterparts. Finally, Section 5.4 proves that the two barbed semi-saturated bisimilarities we introduced capture the barbed congruences proposed so far for mobile ambients, while Section 5.5 applies the framework to the asynchronous CCS.

5.1 Adequacy of IPO Semantics

Several attempts have been made to encode various specification formalisms (Petri nets [50, 62], logic programming [13], CCS [51, 7], λ-calculus [52, 23], asynchronous π-calculus [42], fusion calculus [22], etc.) as reactive systems, either hoping to recover the standard observational equivalences, whenever such a behavioural semantics exists (CCS [48], pi-calculus [49], etc.), or trying to distill a meaningful new semantics, as in the previous chapter for the mobile ambients calculus. The results are however not yet fully satisfactory. On the one-side, IPO-bisimilarity is usually too fine-grained, and mobile ambients are no exception. On the other side, saturated semantics are often too coarse, such as in the case of CCS, where the standard strong bisimilarity is strictly included in the saturated one. The saturated semantics is not indeed able to distinguish certain processes with infinite internal behaviour, hence for example the (recursive) processes $\Omega = rec_z\tau.z$ and $\Theta = \tau.\Omega + a.\Omega$ are saturated bisimilar [54], yet not strong bisimilar [1]. This kind of 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,

[1] In [7], the authors show that the IPO-semantics of CCS coincides with the standard bisimilarity.

can insert them into some context and observe a reduction. However, since in weak semantics reductions cannot be observed, all systems are equivalent.

In the various formalisms this kind of problem has been tackled by using different techniques. Among these, the most famous in the context of process calculi is based on the notion of *barbed bisimulation*, proposed by Milner and Sangiorgi in [54].

5.1.1 Barbed Semantics

Barbed bisimulation represents a general technique for generating bisimulation-based equivalence for any process calculus with a reduction relation and a notion of *barb*. Intuitively, a barb is just a predicate on the states of a system, which simply detects the possibility of performing some observable action. For instance, in Milner's CCS barbs express the ability of a process to perform an input or an output over a channel. Barbed equivalences add the check of such predicates in the bisimulation game: every time that a system shows a barb, the equivalent systems has to show the same barb, and vice-versa.

The advantage of this kind of semantics is that it does not exploit labelled transition systems, and therefore allows us to avoid several labelled transition systems for the same calculus that lead to different behavioral equivalences, such as in the case of the π-calculus [53]. Moreover, the flexibility of the definition allows for recasting a wide variety of observational, bisimulation-based equivalences. For example Milner and Sangiorgi apply their proposal to the CCS, by proving that strong bisimulation of CCS coincides with the congruence induced by barbed bisimulation.

In the following, we fix a family O of barbs, and we write $P\downarrow_o$ if P satisfies $o \in O$.

Definition 5.1 (Barbed Bisimilarity, Barbed Congruence)**.** *A symmetric relation $\mathcal{R}$ is a* barbed bisimulation *if whenever $P \,\mathcal{R}\, Q$ then*

- *if $P\downarrow_o$ then $Q\downarrow_o$;*
- *if $P \to P'$ then $Q \to Q'$ and $P' \,\mathcal{R}\, Q'$.*

Barbed bisimilarity $\sim^B$ *is the largest barbed bisimulation;* barbed congruence $\simeq^B$ *is the largest congruence contained in $\sim^B$.*

Nevertheless, in the setting of reactive systems all efforts have been focussed so far on strong bisimilarity, tackling neither weak nor barbed semantics. So, in the following sections we will introduce a suitable notion of barbed and weak barbed saturated semantics for reactive systems, and their characterization via transition systems labelled with minimal contexts, by exploiting the semi-saturated game.

5.2 Mobile Ambients

This section shortly introduces the strong and weak behavioural equivalences of mobile ambients. We recall to the reader that the calculus has been introduced in Sections 3.1, while in Section 4.1 we introduced its extended version.

We begin by defining barbs for mobile ambients processes. As said in the previous section, a barb o is a predicate over the states of a system, with $P\downarrow_o$ denoting that P satisfies o. In mobile ambients, $P\downarrow_n$ denotes the presence at top-level of a unrestricted ambient n.

Definition 5.2 (Mobile ambients barbs)**.** *Let P be a pure process. It satisfies the* strong barb n*, in symbols $P\downarrow_n$, if $P \equiv (\nu A)(n[Q]|R)$ and $n \notin A$, for some processes Q and R and a set of restricted ambient names A.*

Definition 5.3 (Mobile ambients weak barbs)**.** *Let P be a pure process. It satisfies the* weak barb n*, in symbols $P\Downarrow_n$, if there exists a process P' such that $P \to^* P'$ and $P'\downarrow_n$, where $\to^*$ is the transitive and reflexive closure of $\to$.*

The two notions above are exploited to give the definitions of strong [60] and weak [47] reduction barbed congruence, respectively. Before presenting them, we introduce MAs *contexts*: they are MAs processes with a hole $-$, formally generated by the following grammar (for R mobile ambient process)

$$C[-] ::= -,\ n[C[-]],\ M.C[-],\ (\nu n)C[-],\ C[-] \mid R.$$

Definition 5.4 (Strong reduction barbed congruence)**.** Strong reduction barbed congruence $\sim^{MA}$ *is the largest symmetric relation* $\mathcal{R}$ *such that whenever* $P \,\mathcal{R}\, Q$ *then*

- *if* $P \downarrow_n$ *then* $Q \downarrow_n$*;*
- *if* $P \rightarrow P'$ *then* $Q \rightarrow Q'$ *and* $P' \,\mathcal{R}\, Q'$*;*
- $\forall C[-], C[P] \,\mathcal{R}\, C[Q]$.

Definition 5.5 (Weak reduction barbed congruence)**.** Weak reduction barbed congruence $\sim^{WMA}$ *is the largest symmetric relation* $\mathcal{R}$ *such that whenever* $P \,\mathcal{R}\, Q$ *then*

- *if* $P \downarrow_n$ *then* $Q \Downarrow_n$*;*
- *if* $P \rightarrow P'$ *then* $Q \rightarrow Q'$ *and* $P' \,\mathcal{R}\, Q'$*;*
- $\forall C[-], C[P] \,\mathcal{R}\, C[Q]$.

Labelled characterization of reduction barbed congruences over mobile ambients processes are presented by Rathke and Sobociński for the strong case [60], and by Merro and Zappa Nardelli for the weak one [47].

The main result we will present in this chapter is the proposal of a novel notion of barbed saturated bisimilarity over reactive systems, both for the strong and weak case, that is able to capture the two behavioural semantics for mobile ambients defined above.

5.3 Barbed Semantics for Reactive Systems

This section proposes a notion of *barbed saturated bisimilarity* for reactive systems, showing that it is efficiently characterized through the IPO-transition systems by exploiting the semi-saturated game [6]: Section 5.3.1 studies the strong case; Section 5.3.2, the weak one.

5.3.1 Barbed Saturated Bisimilarity

Barbed congruence introduced in Definition 5.1 is clearly a congruence, but there is no guarantee that it is also a bisimulation. Here we consider a different notion of behavioural equivalence that is both a bisimulation and a congruence.

Definition 5.6 (Barbed saturated bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* barbed saturated bisimulation *if whenever* $P \,\mathcal{R}\, Q$ *then* $\forall C[-]$

- *if* $C[P] \downarrow_o$ *then* $C[Q] \downarrow_o$*;*
- *if* $C[P] \rightarrow P'$ *then* $C[Q] \rightarrow Q'$ *and* $P' \,\mathcal{R}\, Q'$.

Barbed saturated bisimilarity $\sim^{BS}$ *is the largest barbed saturated bisimulation.*

It is easy to see that $\sim^{BS}$ is the largest barbed bisimulation that is also a congruence, and that it is finer than $\simeq^{B}$ (the largest congruence contained into barbed bisimilarity). Intuitively, in the former case the external observer can plug systems into contexts at any step of their execution, while in the latter the observer can contextualize systems only at the beginning. The former observer is more powerful than the latter, thus proving that $\sim^{BS}$ is indeed finer than $\simeq^{B}$.

It is our opinion that $\sim^{BS}$ is more appropriate, in order to model concurrent interactive systems embedded in an environment that continuously changes. And while in several formalisms the two notions coincide [31], for mobile ambients calculus the standard behavioural equivalence $\sim^{MA}$ (Definition 5.4) is clearly an instance of $\sim^{BS}$.

Most importantly, though, barbed saturated bisimilarity can be efficiently characterized through the IPO-transition system via the semi-saturated game.

Definition 5.7 (Barbed semi-saturated bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* barbed semi-saturated bisimulation *if whenever* $P \,\mathcal{R}\, Q$ *then*

- $\forall C[-]$, *if* $C[P] \downarrow_o$ *then* $C[Q] \downarrow_o$*;*
- *if* $P \xrightarrow{C[-]}_{IPO} P'$ *then* $C[Q] \rightarrow Q'$ *and* $P' \,\mathcal{R}\, Q'$*.*

Barbed semi-saturated bisimilarity $\sim^{BSS}$ *is the largest barbed semi-saturated bisimulation.*

Proposition 5.1. In a reactive system having redex-IPOs, $\sim^{BSS} = \sim^{BS}$.

The proof of Proposition 5.1 is shown in Section C.1 (Appendix C).

Reasoning on $\sim^{BSS}$ is easier than on $\sim^{BS}$, because instead of looking at the reductions in all contexts, we consider only IPO-transitions. Even if barbs are still quantified over all contexts, for many formalisms (as for mobile ambients) it is actually enough to check if $P \downarrow_o$ implies $Q \downarrow_o$, since this condition implies that $\forall C[-]$, if $C[P] \downarrow_o$ then $C[Q] \downarrow_o$. Barbs satisfying this property are called *contextual* barbs.

Definition 5.8 (Contextual barbs)**.** *A barb* o *is a* contextual barb *if whenever* $P \downarrow_o$ *implies* $Q \downarrow_o$ *then* $\forall C[-]$, $C[P] \downarrow_o$ *implies* $C[Q] \downarrow_o$*.*

5.3.2 Weak Barbed Saturated Bisimilarity

This section introduces weak barbed (semi-)saturated bisimilarity. We begin by recalling weak barbs. A state P satisfies the weak barb o (written $P \Downarrow_o$) if there exists a state P' such that $P \rightarrow^* P'$ and $P' \downarrow_o$.

Definition 5.9 (Weak barbed saturated bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* weak barbed saturated bisimulation *if whenever* $P \,\mathcal{R}\, Q$ *then* $\forall C[-]$

- *if* $C[P] \Downarrow_o$ *then* $C[Q] \Downarrow_o$*;*
- *if* $C[P] \rightarrow^* P'$ *then* $C[Q] \rightarrow^* Q'$ *and* $P' \,\mathcal{R}\, Q'$*.*

Weak barbed saturated bisimilarity $\sim^{WBS}$ *is the largest weak barbed saturated bisimulation.*

By following the strong case, also weak barbed saturated bisimilarity can be efficiently characterized through the IPO-transition system via the semi-saturated game.

Definition 5.10 (Weak barbed semi-saturated bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* weak barbed semi-saturated bisimulation *if whenever* $P \,\mathcal{R}\, Q$ *then*

- $\forall C[-]$, *if* $C[P] \downarrow_o$ *then* $C[Q] \Downarrow_o$*;*
- *if* $P \xrightarrow{C[-]}_{IPO} P'$ *then* $C[Q] \rightarrow^* Q'$ *and* $P' \,\mathcal{R}\, Q'$*.*

Weak barbed semi-saturated bisimilarity $\sim^{WBSS}$ *is the largest weak barbed semi-saturated bisimulation.*

The correspondence result is stated below.

Proposition 5.2. In a reactive system having redex-IPOs, $\sim^{WBSS} = \sim^{WBS}$.

The proof of Proposition 5.2 is shown in Section C.1 (Appendix C).

Now we introduce weak contextual barbs. Analogously to the strong case, for those formalisms whose barbs are weakly contextual the first condition of Definition 5.10 becomes simpler: indeed, it suffices to check if $P \downarrow_o$ implies $Q \Downarrow_o$.

Definition 5.11 (Weak contextual barbs)**.** *A barb* o *is a* weak contextual barb *if whenever* $P \downarrow_o$ *implies* $Q \Downarrow_o$ *then* $\forall C[-]$, $C[P] \downarrow_o$ *implies* $C[Q] \Downarrow_o$*.*

5.4 Labelled Characterizations of Barbed Congruences for Mobile Ambients

This section proposes a labelled characterization of both strong and weak reduction barbed congruences for mobile ambients, presented in Section 5.2. Indeed, mobile ambients can be seen as a reactive system, with pure processes (up-to structural congruence) as ground terms and with the contexts generated by the following grammar (for R mobile ambient process) $C[-] ::= -, n[C[-]], (\nu n)C[-], C[-] \mid R$ as contexts. As shown in Chapter 3, pure processes must first be encoded into graphs, and the reduction semantics simulated by graph rewriting. We can then apply the *borrowed contexts* technique for distilling IPOs, which is proved to be an instance of the reactive system construction. The resulting ITS is the one that we presented in Section 4.6. Therefore, we can apply the notions of (weak) barbed saturated and semi-saturated bisimilarities, shown in the previous section, in order to capture the two behavioural semantics of mobile ambients.

The first step is stated by the proposition below.

Proposition 5.3. Strong reduction barbed congruence over mobile ambients $\sim^{MA}$ coincides with barbed saturated bisimilarity $\sim^{BS}_{MA}$ for the calculus. Similarly, weak reduction barbed congruence over mobile ambients $\sim^{WMA}$ coincides with weak barbed saturated bisimilarity $\sim^{WBS}_{MA}$ for the calculus.

Note that, in spite of $\sim^{MA}$ and $\sim^{WMA}$ consider more contexts (they consider also contexts of the shape $M.-$) than $\sim^{BS}_{MA}$ and $\sim^{WBS}_{MA}$, respectively, in both cases the correspondence trivially holds. This is due to the fact that processes of the shape $M.P$ have no reduction.

As shown in Section 5.3, we can efficiently characterize (weak) barbed saturated bisimilarity through the IPO-transition system, and the semi-saturated game. We can then characterize strong and weak reduction barbed congruence over mobile ambients by instantiating Definitions 5.7 and 5.10, respectively, with the ITS $\mathcal{D}_\mathcal{I}$ introduced in Section 4.6.

Moreover, the quantification over all contexts can be removed from the first condition of both definitions of strong and weak semi-saturated bisimulation.

Proposition 5.4. Mobile ambients barbs are both strong and weak contextual barbs.

The proof of Proposition 5.4 is shown in Section C.2 (Appendix C).

We then obtain a simpler definition of (weak) semi-saturated bisimilarity.

Definition 5.12 (Barbed semi-saturated bisimulations for mobile ambients)**.** *A symmetric relation $\mathcal{R}$ is a* barbed semi-saturated bisimulation *for mobile ambients if whenever $P \,\mathcal{R}\, Q$ then*

- *if $P \downarrow_n$ then $Q \downarrow_n$;*
- *if $P \xrightarrow{C[-]}_{\mathcal{D}_\mathcal{I}} P'$ then $C[Q] \to Q'$ and $P' \,\mathcal{R}\, Q'$.*

Barbed semi-saturated bisimilarity $\sim^{BSS}_{MA}$ *is the largest barbed semi-saturated bisimulation.*

A symmetric relation $\mathcal{R}$ is a weak barbed semi-saturated bisimulation *for mobile ambients if whenever $P \,\mathcal{R}\, Q$ then*

- *if $P \downarrow_n$ then $Q \Downarrow_n$;*
- *if $P \xrightarrow{C[-]}_{\mathcal{D}_\mathcal{I}} P'$ then $C[Q] \to^* Q'$ and $P' \,\mathcal{R}\, Q'$.*

Weak barbed semi-saturated bisimilarity $\sim^{WBSS}_{MA}$ *is the largest weak barbed semi-saturated bisimulation.*

We finally introduce the main characterization theorem of the chapter.

Theorem 5.1. *Barbed semi-saturated bisimilarity for mobile ambients $\sim^{BSS}_{MA}$ coincides with strong reduction barbed congruence $\sim^{MA}$. Similarly, weak barbed semi-saturated bisimilarity $\sim^{WBSS}_{MA}$ coincides with weak reduction barbed congruence $\sim^{WMA}$.*

It is easy to note that the two statements of the theorem above follow from Proposition 5.3, and from Proposition 5.1 and 5.2, respectively.

5.4.1 On Observing Ambient Migration

An alternative labelled characterization of weak reduction barbed congruence is presented in [47] by Merro and Zappa Nardelli. However, the bisimulation that they propose is not defined in the standard way. They indeed note that in mobile ambients the ability of a (restricted) ambient to migrate is unobservable, therefore in order to take this phenomenon into account they propose a modification of the usual definition of bisimulation. On the contrary, Rathke and Sobociński use instead in [60] the ordinary bisimilarity for characterizing the strong reduction barbed congruence. However, they are forced to add a set of what they call Honda-Tokoro rules, in order to account for the same phenomenon about ambient migrations. We remark that in our proposal we are never able to observe migrations of private ambients, thanks to the use of semi-saturations: this is shown by the following example for the weak semi-saturated case.

Example 5.1. Let us consider the example below, originally proposed in [47], which illustrates two weak reduction barbed congruent processes

$$P = (\nu n)n[in\ k.\mathbf{0}] \qquad \text{and} \qquad Q = \mathbf{0}$$

The two processes P and Q are distinguished by the standard weak equivalence over our LTS $\mathcal{D}_{\mathcal{I}}$, since P can interact with a context $-|k[R]$, while $\mathbf{0}$ cannot. The weak barbed semi-saturated bisimulation instead does not observe the migration of the private ambient n. The transition $P \xrightarrow{-|k[R]}_{\mathcal{D}_{\mathcal{I}}} (\nu n)k[n[0]|R]$ is indeed matched by $0|k[R] \to^* 0|k[R]$. Moreover, since $(\nu n)k[n[0]|R]$ and $0|k[R]$ are weak barbed semi-saturated equivalent, also P and Q are so.

5.5 Labelled Characterizations of Asynchronous Bisimilarity

This section proposes a labelled characterization of the asynchronous bisimilarity for the asynchronous CCS (Section 3.9), by exploiting the IPO LTS presented in Section 4.9. As mobile ambients, asynchronous CCS can indeed be seen as a reactive system, with pure processes (up-to structural congruence) as ground terms and with the contexts generated by the following grammar $C[-] ::= -,\ (\nu n)C[-],\ C[-] \mid R$ (for R ACCS process) as contexts.

We begin by introducing the definition of barb for the asynchronous CCS. The main difference with respect to the synchronous version of the calculus lies in the notion of observation. Since sending messages is non-blocking, an external observer can just send messages to a system without knowing if they will be received or not. For this reason receiving should not be observable and thus barbs take into account only outputs.

Definition 5.13 (ACCS barbs)**.** *Let P be a pure process. It satisfies the* strong barb $\overline{a}$*, in symbols $P \downarrow_{\overline{a}}$, if $P \equiv (\nu A)(\overline{a} \mid Q)$ and $a \notin A$, for some process Q and a set of restricted channel names A.*

Now, the first step is stated by the proposition below [1], which is confirmed by the results presented in Section 6.3.

Proposition 5.5. Asynchronous bisimilarity $\sim^A$ coincides with barbed saturated bisimilarity $\sim^{BS}_A$ for the asynchronous CCS.

We can efficiently characterize barbed saturated bisimilarity through the IPO-transition system, and the semi-saturated game. So, we can characterize asynchronous bisimilarity by instantiating Definitions 5.7, with the ITS A_I (Section 4.9).

Moreover, we can remove the quantification over all contexts from the first condition of the definition of (semi-)saturated bisimulation.

Proposition 5.6. Asynchronous CCS barbs are strong contextual barbs.

The proof of Proposition 5.6 can be obtained by following the one of Proposition 5.4.

We then obtain a simpler definition of semi-saturated bisimilarity.

Definition 5.14 (Barbed semi-Saturated bisimulations for ACCS)**.** *A symmetric relation $\mathcal{R}$ is a* barbed semi-saturated bisimulation *for* ACCS *if whenever $P\,\mathcal{R}\,Q$ then*

- *if* $P \downarrow_{\overline{a}}$ *then* $Q \downarrow_{\overline{a}}$*;*
- *if* $P \xrightarrow{C[-]}_A P'$ *then* $C[Q] \rightarrow Q'$ *and* $P' \,\mathcal{R}\, Q'$.

Barbed semi-saturated bisimilarity $\sim_A^{BSS}$ *is the largest barbed semi-saturated bisimulation.*

Theorem 5.2. *Barbed semi-saturated bisimilarity for* ACCS $\sim_A^{BSS}$ *coincides with asynchronous bisimilarity* $\sim^A$ *(Definition 3.19).*

The theorem above follows from Propositions 5.5 and 5.1.

5.6 Summary

The main issues of this chapter have been the introduction of barbed bisimilarities in reactive systems, and their exploitation for recasting the semantics of mobile ambients and asynchronous CCS.

In particular, we proposed the novel notions of barbed and weak barbed saturated bisimilarity over reactive systems, showing that they can be efficiently characterized through the IPO-transition systems by employing the semi-saturated game. We applied the framework to mobile ambients, proving that it can capture the strong and the weak reduction barbed congruence for the calculus, proposed by Rathke and Sobociński [60], and by Merro and Zappa Nardelli [47], respectively. Moreover, also for asynchronous CCS, we showed that our proposal is able to address the standard semantics of the calculus.

We thus obtained a labelled characterization for the barbed congruences of mobile ambients and the asynchronous bisimilarity, exploiting the two ITSs for these calculi previously proposed in Chapter 4.

As far as the mobile ambients calculus, as discussed in Section 5.4, we recall that an alternative, labelled characterization of the strong reduction barbed congruence is presented in [60]. Rathke and Sobociński use there the standard bisimilarity to capture the congruence, but they are forced to add a set of Honda-Tokoro rules to deal with the unobservability of ambient migrations. Our solution instead accounts for this phenomenon by the use of the barbed semi-saturated bisimulation. It is true however that the proposal in [60] does not need any additional observation, while in our approach the choice of the right notion of barb is left to the ingenuity of the researcher.

Chapter 6

On barbs and labels in reactive systems

In this chapter we move one further step in dealing with the adequacy issue of the standard semantics (the IPO and the saturated one) for reactive systems. In particular, we propose a novel behavioural equivalence for reactive systems, namely, L-bisimulation: a flexible tool, since it is parametric with respect to a set of minimal labels L. Also in this case the idea is very simple, and it just asymmetrically refines the standard bisimulation game. If the minimal LTS has a transition $P \xrightarrow{C[-]} Q$, then a bisimilar P' has to react via a minimal transition $P' \xrightarrow{C[-]} Q'$, whenever $C[-] \in L$; or it must ensure that $C[P']$ may evolve into Q' (thus requiring no minimality for $C[-]$ w.r.t. P'), otherwise. The associated bisimilarity is intermediate between the standard semantics (i.e., minimal and saturated) for reactive systems: indeed, it is able to recover both of them, by simply varying the set L and exploiting the so-called semi-saturated semantics. It can be proved that, under mild closure conditions on the set L, L-bisimilarity is a congruence; and moreover, it can be shown that barbed saturated semantics can be recast, as long as L satisfies suitable barb-capturing properties.

With respect to barbed saturated semantics, L-bisimilarity admits a streamlined definition, where state predicates play no role, so resulting in simpler verification. We test its adequacy and ease of use against suitable case studies. We thus consider the minimal context semantics for mobile ambients and ACCS introduced in Section 4 and we show that in those cases, a set L of minimal labels can be identified, such that L-bisimilarity precisely captures the standard semantics of the calculus at hand.

The paper is organized as follows. Section 6.1 presents the technical core of the chapter: the introduction of L-bisimilarity for reactive systems, the proof that (under mild conditions on L) it is a congruence, and moreover its correspondence with barbed semantics. Finally, Section 6.2 and Section 6.3 prove that, suitably varying the set L, the newly defined L-bisimilarity captures the standard equivalences for mobile ambients and for asynchronous CCS, respectively.

6.1 A New Semantics for Reactive Systems: L-Bisimilarity

As shown [7], in the case of CCS, IPO-bisimilarity coincides with the ordinary bisimilarity. However, for many interesting cases, such as mobile ambients and ACCS (as discussed in Chapter 5), it is often too fine-grained. On the other side, as for CCS, saturated bisimilarity is often too coarse.

In this section we introduce L-indexed bisimilarity (shortly, L-bisimilarity), a novel kind of bisimilarity parametric with respect to a class of contexts (also referred to as *labels*) L. For each class L satisfying some closure properties, the new equivalence $\sim^L$ is a congruence and $\sim^I \subseteq \sim^L \subseteq \sim^S$.

Intuitively, L-bisimulations can be thought as something in between IPO-bisimulations and semi-saturated bisimulations: when $P \xrightarrow{C[-]}_{IPO}$, if $C[-]$ belongs to L, then Q must perform $Q \xrightarrow{C[-]}_{IPO}$ (as in the IPO-bisimulation), otherwise $C[Q] \rightarrow$ (as in the semi-saturated bisimulation).

Definition 6.1 (L-Bisimulation)**.** *Let L be a class of contexts. A symmetric relation $\mathcal{R}$ is an* L-bisimulation *if whenever $P \,\mathcal{R}\, Q$ then*

$$\textit{if } P \xrightarrow{C[-]}_{IPO} P' \textit{ then} \begin{cases} Q \xrightarrow{C[-]}_{IPO} Q' \textit{ and } P' \,\mathfrak{R}\, Q', & \textit{if } C[-] \in L; \\ C[Q] \to Q' \textit{ and } P' \,\mathfrak{R}\, Q', & \textit{otherwise.} \end{cases}$$

L-bisimilarity $\sim^L$ *is the largest* L*-bisimulation.*

It is easy to note that $\sim^L$ generalizes both $\sim^I$ and $\sim^{SS}$ (and thus $\sim^S$). Indeed, in order to characterize the former, it is enough to take as L the whole class of contexts, while to characterize the latter, we take as L the empty class. In Subsection 6.1.1, we will show that for some L, L-bisimilarity also coincides with barbed saturated bisimilarity. In the remainder of this section, we show that $\sim^L$ is a congruence. In order to prove this, we have to require the following condition on L.

Definition 6.2. *Let* L *be a class of arrows of a category. We say that* L *is IPO-closed, if whenever the following diagram is an IPO and* $b \in L$*, then also* $c \in L$*.*

$$\begin{tikzcd}
 & {} & \\
{} \arrow[ur, "b"] & & {} \arrow[ul, "d"'] \\
 & {} \arrow[ul, "a"] \arrow[ur, "c"'] &
\end{tikzcd}$$

It is often hard to prove that a class of contexts is IPO-closed. It becomes easier with concrete instances of reactive systems that supply a constructive definition for IPOs, such as borrowed contexts.

Proposition 6.1. Let us consider a reactive system with redex RPOs and an IPO-closed class L of contexts. Then, $\sim^L$ is a congruence.

$$\begin{tikzcd}
 & k_6 & & \\
k_4 \arrow[ur, "{J[-]}"] & & & \\
 & k_2 \arrow[ul, "{C[-]}"] & & k_3 \arrow[ulul, "{D[-]}"'] \\
 & & 0 \arrow[ul, "P"] \arrow[ur, "L"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
 & k_6 & & \\
k_4 \arrow[ur, "{J[-]}"] & & k_5 \arrow[ul, "{D_2[-]}"'] & \\
 & k_2 \arrow[ul, "{C[-]}"] \arrow[ur, "{J'[-]}"] & & k_3 \arrow[ul, "{D_1[-]}"'] \\
 & & 0 \arrow[ul, "P"] \arrow[ur, "L"'] &
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
 & k_6 & & \\
k_4 \arrow[ur, "{J[-]}"] & & k_5 \arrow[ul, "{D_2[-]}"'] & \\
 & k_2 \arrow[ul, "{C[-]}"] \arrow[ur, "{J'[-]}"] & & k_3 \arrow[ul, "{E[-]}"'] \\
 & & 0 \arrow[ul, "Q"] \arrow[ur, "L'"'] &
\end{tikzcd}$$

(iii)

Proof. In order to prove this theorem we will use the composition and decomposition property of IPOs (Proposition 2.2). We have to prove that if $P \sim^L Q$ then $C[P] \sim^L C[Q]$. We show that $\mathfrak{R} = \{(C[P], C[Q])$ s.t. $P \sim^L Q\}$ is an L-bisimulation.

Suppose that $C[P] \xrightarrow{J[-]}_{IPO} P'$. Then there exists an IPO square like diagram (i) above, where $\langle L, R\rangle \in \mathfrak{R}$, $D[-] \in \mathbf{D}$ and $P' = D[R]$. Since, by hypothesis, the reactive system 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 [45]). 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.

Since $\mathbf{D}$ is composition-reflecting, then both $D_1[-]$ and $D_2[-]$ belong to $\mathbf{D}$ and then $P \xrightarrow{J'[-]}_{IPO} D_1[R]$. Now there are two cases: either $J[-] \in L$ or $J[-] \notin L$.

If $J[-] \in L$, then also $J'[-] \in L$, because L is IPO-closed, by hypothesis. Since $P \sim^L Q$, then $Q \xrightarrow{J'[-]}_{IPO} Q''$ and $D_1[R] \sim^L Q''$. This means that there exists an IPO square like the lower square of diagram (iii) above, where $\langle L', R'\rangle \in \mathfrak{R}$, $E[-] \in \mathbf{D}$ and $E[R] = Q''$. Now recall by the previous observation that the upper square of diagram (iii) is also an IPO and then, by IPO composition, also the outer square is an IPO. This means that $C[Q] \xrightarrow{J[-]}_{IPO} D_2[Q'']$. Since $D_1[R] \sim^L Q''$, then $P' = D[R] = D_2[D_1[R]] \,\mathfrak{R}\, D_2[Q'']$.

If $J[-] \notin L$, then either $J'[-] \in L$ or $J'[-] \notin L$. In both cases, from $P \xrightarrow{J'[-]}_{IPO} D_1[R]$ we derive that $J'[Q] \to Q''$ and $D_1[R] \sim^L Q''$. This means that the lower square of diagram (iii) above commutes. Since also the upper square commutes, then also the outer square commutes. This means that $C[Q] \to D_2[Q'']$. Since $D_1[R] \sim^L Q''$, then $P' = D[R] = D_2[D_1[R]] \,\mathfrak{R}\, D_2[Q'']$. □

6.1.1 Barbed Saturated Bisimilarity via L-bisimilarity

Here we show that L-bisimilarity can also characterize barbed saturated bisimilarity, whenever barbs and the set of labels L satisfies suitable conditions. This result will be used in later sections in order to show that L-bisimilarity captures the correct equivalences for mobile ambients and ACCS.

In order to guarantee that $\sim^L \subseteq \sim^{BS}$, we need some conditions ensuring that the checking of barbs of $\sim^{BS}$ is already done in $\sim^L$ by the labels in L.

Definition 6.3. *Let L be a set of labels and let O be a set of barbs. We say that L is O-*capturing *if for each barb o there exists a label $C[-] \in L$ such that for each process P, $P \downarrow_o$ if and only if $P \xrightarrow{C[-]}_{IPO} P'$.*

The next two definitions are needed to ensure that $\sim^{BS} \subseteq \sim^L$.

Definition 6.4. *Let $\mathcal{R}$ be a relation and let $\mathcal{P}(X, Y)$ be a binary predicate on processes. We say that $\mathcal{P}(X, Y)$ is* stable under $\mathcal{R}$ *if whenever $P\mathcal{R}Q$ and $\mathcal{P}(P, P')$ there exists Q' such that $\mathcal{P}(Q, Q')$ and $P'\mathcal{R}Q'$.*

For example, the predicates in Figure 6.1 and Figure 6.2 are stable under $\sim^{BS}$.

Definition 6.5. *Let $\mathcal{R}$ be a relation and let $C[-]$ be a label. We say that $C[-]$ is* stable under $\mathcal{R}$ *if the predicate $\mathcal{P}(X, Y) = X \xrightarrow{C[-]}_{IPO} Y$ is stable under $\mathcal{R}$.*

Note that the definition above says that the relation $\mathcal{R}$ is a bisimulation for the label $C[-]$. We will use it to ensure that $\sim^{BS}$ is a bisimulation for all the labels in L.

We can finally state a first correspondence result.

Proposition 6.2. Let us consider a reactive system with redex RPOs, a set O of contextual barbs and a set L of labels. If L is O-capturing and all its labels are stable under $\sim^{BS}$, then $\sim^{BS}$ coincides with $\sim^L$.

Proof. In order to prove that $\sim^{BS} \subseteq \sim^L$, we show that $\mathcal{R} = \{(P, Q) \text{ s.t. } P \sim^{BS} Q\}$ is an L-bisimulation.

Suppose that $P \xrightarrow{C[-]}_{IPO} P'$. We have two cases: either $C[-] \in L$ or $C[-] \notin L$. If $C[-] \in L$, then $C[-]$ is stable under $\sim^{BS}$ and thus, since $P \sim^{BS} Q$, $Q \xrightarrow{C[-]}_{IPO} Q'$ and $P' \sim^{BS} Q'$. For the case that $C[-] \notin L$, it is enough to note that, since $P \xrightarrow{C[-]}_{IPO} P'$, then $C[P] \to P'$. Since $P \sim^{BS} Q$, then $C[Q] \to Q'$ and $P' \sim^{BS} Q'$.

Now we show that $\mathcal{R} = \{(P, Q) \text{ s.t. } P \sim^L Q\}$ is a barbed semi-saturated bisimulation (i.e., $\sim^L \subseteq \sim^{BSS}$) and thus, since the reactive system has redex IPOs, by Proposition 5.1 it follows that $\sim^L \subseteq \sim^{BS}$.

At first, we note that, since O is a set of contextual barbs, in order to show that $\mathcal{R}$ satisfies the first condition of Definition 5.7 it suffices to show that $P \downarrow_o$ implies $Q \downarrow_o$. Since L is O-capturing, if $P \downarrow_o$ then there is a label $C[-] \in L$ such that $P \downarrow_o$ if and only if $P \xrightarrow{C[-]}_{IPO}$. Since $P \sim^L Q$, then also $Q \xrightarrow{C[-]}_{IPO}$ and $Q \downarrow_o$.

In order to prove the second condition of Definition 5.7, it is enough to note that if $P \xrightarrow{C[-]}_{IPO} P'$ then, in both the case that $C[-] \in L$ and $C[-] \notin L$, $C[Q] \to Q'$ with $P' \sim^L Q'$. □

As a corollary of the previous definition, we obtain the following property that allows to check whenever IPO-bisimilarity coincides with barbed saturated one.

Lemma 6.1. *Let us consider a reactive system with redex IPOs. If the barbs are contextual, the set of all labels is O-capturing, and each label is stable under $\sim^{BS}$, then $\sim^I$ coincides with $\sim^{BS}$.*

6.2 L-Bisimilarity for Mobile Ambients

This section proposes a new labelled characterization of the reduction barbed congruence for mobile ambients, presented in Section 5.2 (Definition 5.4). In particular, by using the IPO LTS $\mathcal{D}_{\mathcal{I}}$ (in Section 4.6) we define an L-bisimilarity that captures barbed saturated bisimilarity for mobile ambients, coinciding with reduction barbed congruence as shown in Section 5.4 (Proposition 5.3).

As discussed in Section 6.1.1, we can characterize barbed saturated bisimilarity on a set of contextual barbs O through the IPO transition system and a set of labels L. In particular, as required by Proposition 6.2, the set L must be O-capturing and each $C[-] \in L$ must be stable under the barbed saturated bisimilarity.

We denote by O_{MA} the set of barbs of mobile ambients, recalling that mobile ambients barbs are contextual barbs (Proposition 5.4).

Proposition 6.3. O_{MA} is a set of contextual barbs.

Therefore, we can characterize reduction barbed congruence over mobile ambients by instantiating Definitions 6.1 with the IPO LTS $\mathcal{D}_{\mathcal{I}}$ and a set L of labels having the two properties said above.

First of all, we find some labels of $\mathcal{D}_{\mathcal{I}}$ that capture the barbs of mobile ambients. This ensures that the checking of barbs of the barbed saturated bisimilarity is done in the L-bisimilarity by the first condition of its definition. It is easy to note that a mobile ambients process P observes a unrestricted ambient n at top-level, in symbols $P \downarrow_n$, if and only if it can execute a transition labelled with $- \mid \mathit{open}\ n.T_1$ or with $- \mid m[\mathit{in}\ n.T_1 \mid T_2]$. Therefore, L is O_{MA}-capturing if it contains at least one kind of these labels. We choose to consider labels of the first type, that is, having the shape $- \mid \mathit{open}\ n.T_1$, for n ambient name and T_1 pure process.

It is possible to prove that these labels are stable under $\sim^{BS}_{MA}$. Therefore, if we consider the set L defined below, we obtain an L-bisimilarity for mobile ambients that is able to characterize $\sim^{BS}_{MA}$.

Proposition 6.4. Let L_{MA} be the set of all labels of the ITS $\mathcal{D}_I$ having the shape $- \mid \mathit{open}\ n.T_1$, for n ambient name and T_1 pure process. Then, L_{MA} is O_{MA}-capturing.

Proof. We have to show that for each barb $n \in O_{MA}$ there exists a label $C[-] \in L_{MA}$ such that for each process P, $P \downarrow_n$ if and only if $P \xrightarrow{C[-]}_{\mathcal{D}_{\mathcal{I}}} P'$.

It is easy to note that, given a barb $n \in O_{MA}$, we have that for each process P, $P \downarrow_n$ if and only if $P \xrightarrow{-|\mathit{open}\ n.T_1}_{\mathcal{D}_{\mathcal{I}}} P'$, with T_1 pure process. Since we know that L_{MA} contains all labels having the shape $- \mid \mathit{open}\ n.T_1$, for n ambient name and T_1 pure process, we can conclude that L_{MA} is O_{MA}-capturing. □

Now, in order to prove that each $C[-] \in L_{MA}$ is stable under $\sim^{BS}_{MA}$, we exploit a predicate such that it is stable under $\sim^{BS}_{MA}$ and equivalent to the one of Definition 6.5. More explicitly, we will prove that the predicate in Figure 6.1 coincides with $\mathcal{P}(X, Y) = X \xrightarrow{-|\mathit{open}\ n.T_1}_{\mathcal{D}_{\mathcal{I}}} Y$. Indeed, the fact that $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ holds, means that P inside the context $C'[-]$ can evolve into P'' that observe m, and since m is fresh, it means that the capability $\mathit{open}\, n$ has been performed. Moreover, the condition on P' ensures that the resulting states of the two predicates coincide.

Lemma 6.2. *Let $\mathcal{P}^{-|\mathit{open}\ n.T_1}(X, Y)$ be the binary predicate on mobile ambients processes shown in Figure 6.1, for n ambient name and T_1 pure process. Then, $\mathcal{P}^{-|\mathit{open}\ n.T_1}(X, Y)$ is stable under $\sim^{BS}_{MA}$ and for each P and P', $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ if and only if $P \xrightarrow{-|\mathit{open}\ n.T_1}_{\mathcal{D}_{\mathcal{I}}} P'$.*

Proof. We begin by proving that the predicate $\mathcal{P}^{-|\mathit{open}\ n.T_1}(X, Y)$ is stable under $\sim^{BS}_{MA}$.

Assume that $P \sim^{BS}_{MA} Q$ and $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ holds. Since $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ holds, then there exists a process P'' and an ambient m fresh for P and Q, such that $C'[P] \to P''$, $P'' \downarrow_m$, $P'' \to P'$ and $P' \not\downarrow_m$, with $C'[-] = - \mid \mathit{open}\ n.(m[\mathbf{0}] \mid \mathit{open}\ m.T_1)$.

Since $C'[P] \to P''$ and $P \sim^{BS}_{MA} Q$, then $C'[Q] \to Q''$ and $P'' \sim^{BS}_{MA} Q''$. Therefore, it is obvious that also $Q'' \downarrow_m$. Now, we know that $P'' \to P'$, hence we can say that $Q'' \to Q'$ and $P' \sim^{BS}_{MA} Q'$. From this follows that, since $P' \not\downarrow_m$, then also $Q' \not\downarrow_m$. So, we can conclude that $\mathcal{P}^{-|\mathit{open}\ n.T_1}(Q, Q')$ holds, hence $\mathcal{P}^{-|\mathit{open}\ n.T_1}(X, Y)$ is stable under $\mathcal{R}$.

Now we show that for each P and P', $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ iff $P \xrightarrow{-|\mathit{open}\ n.T_1}_{\mathcal{D}_{\mathcal{I}}} P'$.

Assume that $\mathcal{P}^{-|\mathit{open}\ n.T_1}(P, P')$ holds. This means that there exists a process P'' and an ambient m fresh for P, such that $C'[P] \to P''$, $P'' \downarrow_m$, $P'' \to P'$ and $P' \not\downarrow_m$, with $C'[-] = - \mid \mathit{open}\ n.(m[\mathbf{0}] \mid \mathit{open}\ m.T_1)$. The fact that $C'[P] \to P''$ and $P'' \downarrow_m$ means that the capability $\mathit{open}\ n$ has been executed, hence there must be a unrestricted ambient n at top-level of P, i.e., $P \equiv (\nu A)(n[P_1] \mid P_2)$ and $n \notin A$. From this follows that $P'' = (\nu A)(P_1 \mid P_2) \mid m[\mathbf{0}] \mid \mathit{open}\ m.T_1$, and since $P' \not\downarrow_m$, then $P' \equiv (\nu A)(P_1 \mid P_2) \mid T_1$. Moreover, by knowing that $P = (\nu A)(n[P_1] \mid P_2)$ and $n \notin A$, we can conclude that $P \xrightarrow{-|\mathit{open}\ n.T_1}_{\mathcal{D}_{\mathcal{I}}} P'$.

Assume that $P \xrightarrow{-|\mathit{open}\ n.T_1} P'$. This means that $P \equiv Q$, where $Q = (\nu A)(n[P_1] \mid P_2)$, $n \notin A$ and $P' = (\nu A)(P_1 \mid P_2) \mid T_1$. We consider the context $C'[-] = - \mid \mathit{open}\ n.(m[\mathbf{0}] \mid \mathit{open}\ m.T_1)$ with

$m \notin fn(P)$. It is easy to note that $C'[Q] \to P''$ s.t. $P'' = (\nu A)(P_1 \mid P_2) \mid m[\mathbf{0}] \mid open\ m.T_1$ and $P'' \downarrow_m$. Therefore, since $C'[P] \equiv C'[Q]$, we also have that $C'[P] \to P''$. Now, we can note that $P'' \to P'$ and, since m is fresh for P, $P' \not\downarrow_m$. $\square$

Proposition 6.5. All labels in L_{MA} are stable under $\sim^{BS}_{MA}$.

The proof of the proposition above trivially follows from Lemma 6.2.

We finally introduce the main characterization proposition.

Proposition 6.6. $\sim^{BS}_{MA} = \sim^{L_{MA}}$.

Proof. First of all, by Proposition 6.3, we know that mobile ambients barbs are contextual. Moreover, by Propositions 6.4 and 6.5, we know that L is O_{MA}-capturing and it contains only labels stable under $\sim^{BS}_{MA}$. Therefore, thanks to Proposition 6.2, we can conclude that $\sim^{BS}_{MA} = \sim^{L_{MA}}$. $\square$

The L-bisimilarity $\sim^{L_{MA}}$ presented above is not the only one which is able to characterize barbed saturated bisimilarity $\sim^{BS}_{MA}$. For example, as said before, we can choose to consider all labels of the shape $- \mid m[in\ n.T_1 \mid T_2]$, which besides being able to capture mobile ambients barbs, they are also stable under $\sim^{BS}_{MA}$. However, generally, we can consider the sets L containing at least all the labels of the shape $- \mid open\ n.T_1$ or $- \mid m[in\ n.T_1 \mid T_2]$ to capture barbs, and other labels of $\mathcal{D}_{\mathcal{I}}$ that are stable under $\sim^{BS}_{MA}$, i.e., labels such that it is possible to define a predicate analogous to the one we defined for the labels $- \mid open\ n.T_1$.

6.3 L-Bisimilarity for Asynchronous CCS

In this section we first show that L-bisimilarity is able to capture the standard semantics of asynchronous CCS and then we prove that it also coincides with its barbed saturated bisimilarity.

L-Bisimilarity for Asynchronous CCS. In asynchronous bisimulation (Definition 3.19), transitions labelled with τ and $\bar{a}$ (corresponding to $-$ and $- \mid a.T_1$ in A_I, respectively) must be matched by transitions with the same labels. Moreover, when $P \xrightarrow{a} P'$ (corresponding to $P \xrightarrow{-\mid\bar{a}} P'$ in A_I) then either $Q \xrightarrow{a} Q'$ and $P' \mathcal{R} Q'$ or $Q \xrightarrow{\tau} Q'$ and $P' \mathcal{R} Q' \mid \bar{a}$. This is equivalent to require that $Q \mid \bar{a} \to Q'$ and $P' \mathcal{R} Q'$. Thus, in order to characterize $\sim^{A}$ as L-bisimilarity, it suffices to choose as L the set of labels corresponding to τ and $\bar{a}$.

Proposition 6.7. Let L_A be the set containing the labels of the ITS A_I of the shape $-$ and $- \mid a.T_1$, for $a \in \mathcal{N}$ and $T_1 \in \mathcal{P}$. Then, $\sim^{L_A} = \sim^{A}$.

From L-Bisimilarity to Barbed Saturated Bisimilarity. It is important to note that the choice of L_A is not arbitrary. Indeed, $\sim^{L_A}$ coincides with the barbed saturated bisimilarity for the asynchronous CCS. This is not a new result, but it is interesting to see that it can be easily proved by following the same approach that we have used for mobile ambients in Section 6.2.

Recall that L_A only contains labels of the form $-$ and $- \mid a.T_1$ (corresponding to labels τ and $\bar{a}$ in the ordinary LTS). Since only output barbs $\downarrow_{\bar{a}}$ are defined, then L_A is barb capturing.

We also know that these barbs are contextual, hence, in order to use Proposition 6.2, we only have to prove that all the labels in L_A are stable under barbed congruence. Analogously to mobile ambients, we define some additional predicates. These are shown in Figure 6.2. It is easy to see that for each label $C[-]$ in L_A, $X \xrightarrow{C[-]} Y$ in A_I if and only if $\mathcal{P}^{C[-]}(X, Y)$. It is also easy to show that all of them are stable under $\sim^{BS}$.

$\mathcal{P}^{-\mid open\ n.T_1}(X, Y)$	$\exists P''$ and $m \notin fn(X)$ s.t. $P'' \downarrow_m, C'[X] \to P'' \to Y$ and $Y \not\downarrow_m$ with $C'[-] = - \mid open\ n.(m[\mathbf{0}] \mid open\ m.T_1)$

Figure 6.1: Predicate for the label $- \mid open\ n.T_1$.

$\mathcal{P}^{-\lvert a.T_1}(X,Y)$	$\exists P'$ and $i \notin fn(X)$ s.t. $P' \downarrow \bar{i}$ and $X \lvert a.(\bar{i} \lvert T_1) \lvert i \rightarrow P' \rightarrow Y \not\downarrow i$
$\mathcal{P}^{-}(X,Y)$	$X \rightarrow Y$

Figure 6.2: Predicates for ACCS

Note that the labels of the form $- \mid \bar{a}$ are not stable under $\sim^{BS}$. It is indeed impossible to define a predicate analogous to the ones in Figure 6.2 for $- \mid a.T_1$, since outputs have no continuation.

6.4 Summary

The chapter introduces a novel behavioural equivalence for reactive systems, namely, L-bisimulation: a flexible tool, since it is parametric with respect to a set of labels L. The associated bisimilarity is proved to be a congruence, and it is shown to be intermediate between the standard IPO and saturated semantics for reactive systems: indeed, it is able to recover both of them, by simply varying the set of labels L. More importantly, also the more expressive barbed semantics can be recast, as long as the set L satisfies suitable conditions.

As for any newly proposed semantics, we tested its expressiveness and ease of use against suitable case studies, by using again the mobile ambients and ACCS. We thus considered the IPO transition systems for these calculi proposed in Chapter 4. We showed that in both those cases, for a right choice of L, L-bisimilarity precisely captures the standard semantics for the calculus at hand.

Chapter 7

Conclusions and Future Work

This thesis tackles some issues concerning the adequacy of the standard semantics (IPO and saturated ones) of reactive systems [45], in modelling the concrete semantics of actual formalisms. As discussed in the introduction, the problem is that IPO-bisimilarity is often too fine-grained, as we showed for mobile ambients and asynchronous CCS, while the saturated one may be too coarse, as in the case of synchronous CCS.

Theoretically, one of the main contributions of our work is the introduction of a more expressive semantics for reactive systems which, thanks to its flexibility, allows for recasting a wide variety of observational, bisimulation-based equivalences. In particular, we propose suitable notions of barbed and weak barbed semantics for reactive systems, and their efficient characterization through the IPO-transition systems by exploiting the semi-saturated game.

Another contribution of this thesis is the introduction of a novel, more general behavioural equivalence for reactive systems, namely, L-bisimulation, which is parametric with respect to a set of minimal labels L. We proved that under mild conditions on L the equivalence is a congruence, and most importantly, it is shown to be intermediate between the standard IPO and saturated semantics for reactive system, recasting both of them by varying the set of labels L. Moreover, also the barbed semantics can be recast, as long as the set L satisfies suitable conditions. With respect to barbed semantics, L-bisimilarity is of simpler verification: It indeed admits a streamlined definition, where states predicates play no role.

In order to test the adequacy of our proposals, we instantiated them over the asynchronous CCS and, most importantly, over the calculus of mobile ambients, whose observational semantics is still in a flux. To this end, for each of these two calculi, we described a minimal context semantics, distilled by means of a graphical encoding of the calculus.

The approach we pursued to derive the two LTSs is quite straightforward: for each calculus we proposed a graphical encoding (over standard graphs) such that process congruence is preserved, and we captured the reduction semantics by a set of graph transformation rules, specified using the DPO approach. An IPO-LTS on (processes encoded as) graphs is thus immediately distilled, by applying the borrowed contexts technique, which is an instance of the theory of reactive systems. The derived LTS is then used to define one over processes. So, as far as the LTS for mobile ambients is concerned, it resulted pivotal in proving one of our main practical results, namely, that barbed and weak barbed semantics for mobile ambients do capture the strong [60] and weak [47] barbed congruences for the calculus. Similarly, the LTS for the asynchronous CCS was used to show that strong barbed semantics coincides with the standard semantics of the calculus. We also showed that in both cases, a set L of minimal labels can be identified, such that the resulting L-bisimilarity precisely captures the standard semantics of the calculus at hand.

We do believe that the work presented in this thesis can be considered relevant for the theory of reactive systems for different reasons. First of all, it addresses the adequacy issue for reactive systems, showing the shortcomings of the standard definitions, providing a framework for recasting (weak) barbed equivalence in the framework, and finally proposing a more general semantics, namely L-bisimulation. Moreover, it offers a relevant application for the reactive systems formalism, by applying it to a full-fledged calculus, at the same time showing how borrowed contexts rewriting (an instance of the theory) can be quite effective in deriving LTSs.

We can foresee at least two further extensions of our work.

As far as (weak) barbed equivalence is concerned, we showed that the framework is general enough to capture the abstract semantics of important formalisms such as mobile ambients and asynchronous CCS. However, it is parametric with respect to the choice of the set of barbs and defining the "right" barbs is not a trivial task, as witnessed by several papers about this topic e.g. [58, 40]. So, it would be interesting to extend our framework by considering an automatically derived notion of barb for reactive systems. In [12], authors introduce barbs for adhesive rewriting systems, trying to keep in line with the constructive nature of the borrowed contexts mechanism. To this end, their intuition is driven by the graphical encodings of calculi, and by the nature of barbs in most examples from that setting, there basically (a) barbs check the presence of a suitable subsystem, (b) such that it is needed to perform an interaction with the environment. For instance, in asynchronous CCS, barbs are parallel outputs [1], formally (a) $P \downarrow_{\overline{a}}$ if $P \equiv P_1 \mid \overline{a}$ and (b) these outputs can interact with the environment through the rule $\overline{a} \mid a.Q + M \rightarrow Q$. In mobile ambients, barbs are ambients at the topmost level [47], formally (a) $P \downarrow_m$ if $P \equiv m[P_1] \mid P_2$ and (b) these ambients can be interact with the environment via the rule $open\, m.Q_1 \mid m[Q_2] \rightarrow Q_1 \mid Q_2$. So, a simple notion of barb for adhesive rewriting systems is given: a barb for a system G is defined as a subsystem occurring in it, also occurring in the left-hand-side of some rewriting rule. We do believe that this general mechanism to define barbs for adhesive rewriting systems may help us to solve the problem of automatically identify suitable barbs for reactive systems, along the line of the solution proposed in [38] for bigraphical reactive systems.

As far as L-bisimulation is concerned, first of all, we would like to precisely understand the notion of IPO-closedness, which is required for the set of labels L, in order for L-bisimilarity to be a congruence. We would like to establish suitable and more manageable conditions under which a set of arrows of a given category satisfies that property, especially for those reactive systems where IPOs have an inductive presentation (such as for those induced by the borrowed context mechanism). In more general terms, it would be interesting to further elaborate on the connection between L-bisimilarity and barbed semantics, moving after the preliminary results presented in Section 6.1.1. As a start, in order to establish conditions ensuring that barbs satisfy the pivotal property of being contextual; and, more to the point, for checking whenever a set of labels is barb capturing and contains only labels stable under barbed bisimilarity. As far as the specific case study of mobile ambients is concerned, most of the IPO labels occurring in our transition system are indeed stable, i.e., the relative labelled transitions can be characterized by a predicate which is stable under the barbed saturated bisimilarity. The only labels that are not stable are those ones of the shape $-|m[P]$ and $m[-|P]$ of the rule INAMB and OUTAMB (Figure 4.18), respectively. It seems intriguing that those same labels required the introduction of so-called Honda-Tokoro inference rules in [60] for capturing the reduction barbed congruence by means of standard bisimilarity.

Appendix A

Proofs of Chapter 3

A.1 Mobile Ambients Congruence $\equiv$ versus Graph Isomorphism

In this section we show the proof of Theorem 3.1, which formalises the relation between the structural congruence $\equiv$ and the encoding introduced in Definition 3.16. In order to show the correspondence, we first prove the soundness of the encoding with respect to the structural congruence (Proposition A.2), and then we prove the completeness (Proposition A.4).

We begin by recalling that two processes that are structural congruent have the same free names, as stated by the proposition below.

Proposition A.1. Let P, Q be processes. If $P \equiv Q$, then $fn(P) = fn(Q)$.

The proposition below states the soundness result.

Proposition A.2. Let P, Q be processes and let Γ be a set of names, such that $fn(P) \subseteq \Gamma$. If $P \equiv Q$, then $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$.

Proof. We proceed by induction on the depth of the inference of $P \equiv Q$. Since the proof is straightforward for the laws stating that the parallel operator is associative, commutative and with identity, we do no tackle these cases.

- Suppose that $P \equiv Q$ by the *Cong-Res-Res* rule. It means that $P = (\nu n)(\nu m)P_1$ and $Q = (\nu m)(\nu n)P_1$. If $n = m$, then $P = Q$ and their encodings are obviously the same. Vice versa, if $n \neq m$, then by definition $[\![P]\!]_\Gamma = \{\nu_r \otimes \{(\nu_s \otimes [\![P_1\{r/n\}\{s/m\}]\!]_{\Gamma\cup\{r,s\}}) \circ (0_s \otimes id_{\Gamma\cup\{r\}})\}\} \circ (0_r \otimes id_\Gamma)$, for $r, s \notin \Gamma$ and $r \neq s$. We notice that the value of the last expression is isomorphic to the value of $(\nu_r \otimes \nu_s \otimes [\![P_1\{r/n\}\{s/m\}]\!]_{\Gamma\cup\{r,s\}}) \circ (0_s \otimes 0_r \otimes id_\Gamma)$. Since $r, s \notin \Gamma$ and $r \neq s$, we can write the encoding $[\![Q]\!]_\Gamma$ of Q as $\{\nu_s \otimes \{(\nu_r \otimes [\![P_1\{s/m\}\{r/n\}]\!]_{\Gamma\cup\{s,r\}}) \circ (0_r \otimes id_{\Gamma\cup\{s\}})\}\} \circ (0_s \otimes id_\Gamma)$. Now, the value of the latter expression is isomorphic to the value of $(\nu_s \otimes \nu_r \otimes [\![P_1\{s/m\}\{r/n\}]\!]_{\Gamma\cup\{s,r\}}) \circ (0_r \otimes 0_s \otimes id_\Gamma)$. Moreover, we notice that $P_1\{r/n\}\{s/m\} = P_1\{s/m\}\{r/n\}$, hence, $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$ holds.
- Suppose that $P \equiv Q$ by the *Cong-Res-Par* rule. It means that $P = (\nu n)(P_1 \mid P_2)$, $Q = P_1 \mid (\nu n)P_2$ and $n \notin fn(P_1)$. By definition, $[\![P]\!]_\Gamma = \{\nu_m \otimes ([\![P_1\{m/n\}]\!]_{\Gamma\cup\{m\}} \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}})\} \circ (0_m \otimes id_\Gamma)$, for $m \notin \Gamma$. Since $n \notin fn(P_1)$, we have $P_1\{m/n\} = P_1$, and so $[\![P_1\{m/n\}]\!]_{\Gamma\cup\{m\}} = [\![P_1]\!]_{\Gamma\cup\{m\}}$. Now, we notice that the value of $\nu_m \otimes ([\![P_1]\!]_{\Gamma\cup\{m\}} \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}})$ is isomorphic to the value of $[\![P_1]\!]_\Gamma \otimes (\nu_m \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}})$. We also note that the graph represented by this last expression has the same output interface of the graph represented by $\nu_m \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}}$, hence the sequential composition $(\nu_m \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}}) \circ (0_m \otimes id_\Gamma)$ is defined. Moreover, since m is not in the output interface of $[\![P_1]\!]_\Gamma$, we can easily see that the value of $\{[\![P_1]\!]_\Gamma \otimes (\nu_m \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}})\} \circ (0_m \otimes id_\Gamma)$ is isomorphic to the value of $[\![P_1]\!]_\Gamma \otimes \{(\nu_m \otimes [\![P_2\{m/n\}]\!]_{\Gamma\cup\{m\}}) \circ (0_m \otimes id_\Gamma)\}$. Since the value of the latter expression is also isomorphic to the value of graphical encoding $[\![Q]\!]_\Gamma$, then we can conclude that $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$.

- Suppose that $P \equiv Q$ by the *Cong-Res-Amb* rule. This means that $P = (\nu n)m[P_1]$, $Q = m[(\nu n)P_1]$ and $n \neq m$. By definition, $[\![P]\!]_\Gamma = \{\nu_r \otimes [amb_m \circ (id_m \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}})]\} \circ (0_r \otimes id_\Gamma)$ and $[\![Q]\!]_\Gamma = amb_m \circ \{id_m \otimes [(\nu_r \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}}) \circ (0_r \otimes id_\Gamma)]\}$, for $r \notin \Gamma$. Now, the value of the latter expression is isomorphic to the value of $amb_m \circ \{(id_m \otimes \nu_r \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}}) \circ (0_r \otimes id_\Gamma)\}$, which thanks to the associativity of $\circ$ is isomorphic to $[amb_m \circ (id_m \otimes \nu_r \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}})] \circ (0_r \otimes id_\Gamma)$. Since the operator ν_r is linked to a $\diamond$ node that is both in the output and in the input interface of the graph amb_m, we can conclude that $[\![Q]\!]_\Gamma = \{\nu_r \otimes [amb_m \circ (id_m \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}})]\} \circ (0_r \otimes id_\Gamma) = [\![P]\!]_\Gamma$.

- Suppose that P and Q are α-equivalent. It means that $P = (\nu n)P_1$, $Q = (\nu m)P_1\{m/n\}$ and $m \notin fn(P_1)$. By construction, $[\![P]\!]_\Gamma = (\nu_r \otimes [\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}}) \circ (\mathbf{0}_r \otimes id_\Gamma)$, while $[\![Q]\!]_\Gamma = (\nu_r \otimes [\![P_1\{m/n, r/m\}]\!]_{\Gamma\cup\{r\}}) \circ (\mathbf{0}_r \otimes id_\Gamma)$, for $r \notin \Gamma$. Now, we notice that, since $P_1\{r/n\} = P_1\{m/n, r/m\}$, then $[\![P_1\{r/n\}]\!]_{\Gamma\cup\{r\}} = [\![P_1\{m/n, r/m\}]\!]_{\Gamma\cup\{r\}}$, and therefore $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$.

□

The completeness of our encoding with respect to the structural congruence $\equiv$ is more difficult to prove. So, we need to introduce some additional lemmas. The following lemma allows us to restrict attention to encodings with respect to the set of free names of a process.

Lemma A.1. *Let P be a process, and let Γ be a set of names, such that $fn(P) \subseteq \Gamma$. Then, $[\![P]\!]_\Gamma = [\![P]\!]_{fn(P)} \otimes free_\Gamma$.*

Proof. The proof proceeds by induction on the structure of P.

- Suppose that $P = \mathbf{0}$. By definition, we have $[\![\mathbf{0}]\!]_\Gamma = \mathbf{0}_{a,p} \otimes free_\Gamma$. Since $[\![\mathbf{0}]\!]_\emptyset = \mathbf{0}_{a,p}$, it is immediate to see that $[\![\mathbf{0}]\!]_\Gamma = [\![\mathbf{0}]\!]_\emptyset \otimes free_\Gamma$.

- The cases $P = n[P_1]$ and $P = M.P_1$ are similar. As an example, we consider $P = n[P_1]$. By definition, we have $[\![P]\!]_\Gamma = amb_n \circ (id_n \otimes [\![P_1]\!]_\Gamma)$. Since, by induction hypothesis, $[\![P_1]\!]_\Gamma = [\![P_1]\!]_{fn(P_1)} \otimes free_\Gamma$, then we have $[\![P]\!]_\Gamma = amb_n \circ (id_n \otimes [\![P_1]\!]_{fn(P_1)} \otimes free_\Gamma)$. Now we notice that the value of the latter expression is isomorphic to the value of $[amb_n \circ (id_n \otimes [\![P_1]\!]_{fn(P)})] \otimes free_\Gamma$. Since by definition, we have $[\![P]\!]_{fn(P)} = amb_n \circ (id_n \otimes [\![P_1]\!]_{fn(P)})$, it is easy to see that $[\![P]\!]_\Gamma = [\![P]\!]_{fn(P)} \otimes free_\Gamma$.

- Suppose that $P = (\nu n)P_1$. By definition, we have $[\![(\nu n)P_1]\!]_\Gamma = (\nu_m \otimes [\![P_1\{m/n\}]\!]_{\Gamma\cup\{m\}}) \circ (\mathbf{0}_m \otimes id_\Gamma)$, for $m \notin \Gamma$. Since by induction hypothesis, $[\![P_1\{m/n\}]\!]_{\Gamma\cup\{m\}} = [\![P_1\{m/n\}]\!]_{fn(P_1\{m/n\})} \otimes free_{\Gamma\cup\{m\}}$, then we have $[\![(\nu n)P_1]\!]_\Gamma = (\nu_m \otimes [\![P_1\{m/n\}]\!]_{fn(P_1\{m/n\})} \otimes free_{\Gamma\cup\{m\}}) \circ (\mathbf{0}_m \otimes id_\Gamma)$. Now, we notice that the value of the latter expression is isomorphic to the value of $\{(\nu_m \otimes [\![P_1\{m/n\}]\!]_{fn(P_1\{m/n\})}) \circ (\mathbf{0}_m \otimes id_{fn(P)})\} \otimes free_\Gamma$. Moreover, since $m \notin \Gamma$, then $m \notin fn(P)$, hence, by definition, we have $[\![(\nu n)P_1]\!]_{fn(P)} = (\nu_m \otimes [\![P_1\{m/n\}]\!]_{fn(P)\cup\{m\}}) \circ (\mathbf{0}_m \otimes id_{fn(P)})$. So, since $fn(P_1\{m/n\}) = fn(P) \cup \{m\}$, then we can easily conclude that $[\![P]\!]_\Gamma = [\![P]\!]_{fn(P)} \otimes free_\Gamma$.

- Suppose that $P = P_1 \mid P_2$. By definition, we have $[\![P]\!]_\Gamma = [\![P_1]\!]_\Gamma \otimes [\![P_2]\!]_\Gamma$. Moreover, by induction hypothesis, $[\![P_1]\!]_\Gamma = [\![P_1]\!]_{fn(P_1)} \otimes free_\Gamma$, and analogously $[\![P_2]\!]_\Gamma = [\![P_2]\!]_{fn(P_2)} \otimes free_\Gamma$. So, we have $[\![P]\!]_\Gamma = ([\![P_1]\!]_{fn(P_1)} \otimes free_\Gamma) \otimes ([\![P_2]\!]_{fn(P_2)} \otimes free_\Gamma)$. Thanks to the commutativity and the associativity of $\otimes$, we obtain the graph expression $[\![P]\!]_\Gamma = ([\![P_1]\!]_{fn(P_1)} \otimes [\![P_2]\!]_{fn(P_2)}) \otimes (free_\Gamma \otimes free_\Gamma)$. Now, we notice that $free_\Gamma \otimes free_\Gamma = free_\Gamma$, and since, by definition, $[\![P]\!]_{fn(P)} = [\![P_1]\!]_{fn(P)} \otimes [\![P_2]\!]_{fn(P)}$, we can easily conclude that $[\![P]\!]_\Gamma = [\![P]\!]_{fn(P)} \otimes free_\Gamma$, thanks to the induction hypothesis and to the hypothesis that $fn(P) \subseteq \Gamma$.

□

To prove the completeness result, we need to introduce a normal form for processes. First, for a set of names $N = \{n_1, \ldots, n_z\}$ such that all n_i's are pairwise distinct, let us (νN) denote a shorthand for the composition $(\nu n_1) \ldots (\nu n_k)$.

Proposition A.3 (Normal forms)**.** Let P be a process. There exists a set of names N_r, N_g and a process $nf(P)$, the normal form of P, such that $P \equiv nf(P)$ and the process $nf(P)$ has the shape $((\nu N_r)S) \mid ((\nu N_g)\mathbf{0})$, for $S = m_1[A_1] \mid \ldots \mid m_p[A_p] \mid M_1.B_1 \mid \ldots \mid M_q.B_q$ and such that all A_i's and B_j's are in normal form (yet A_i's have no restrictions at top level) and $N_r \subseteq fn(S)$.

Lemma A.2. *Let P, Q be processes. If $[\![P]\!]_{fn(P)} = [\![Q]\!]_{fn(Q)}$, then $P \equiv Q$.*

Proof. Let P' and Q' be the normal forms of P and Q, respectively. Note that, since $P \equiv P'$ and $Q \equiv Q'$, thanks to the soundness of our encoding and to the hypothesis $[\![P]\!]_{fn(P)} = [\![Q]\!]_{fn(Q)}$, we have that $[\![P']\!]_{fn(P')}$ and $[\![Q']\!]_{fn(Q')}$ denote isomorphic graphs.

The proof proceeds by induction on the structure of P'.

- Suppose that $P' = \mathbf{0}$. By definition, we have $[\![P']\!]_{fn(P')} = \mathbf{0}_{a,p} \otimes free_{fn(P')}$. Since, $[\![P']\!]_{fn(P')}$ and $[\![Q']\!]_{fn(Q')}$ denote isomorphic graphs, they have the same interfaces, hence $fn(P') = fn(Q')$. Furthermore, there must be a bijective correspondence between the set of edges attached to the image of the input p of the graph $[\![P']\!]_{fn(P')}$, and the set of edges attached to the image of the input p of the graph $[\![Q']\!]_{fn(Q')}$. Analogously, there must be a bijective correspondence between the sets of edges attached to the image of the inputs a of both graphs. So, since in $[\![P']\!]_{fn(P')}$ both sets of edges are empty, it is obvious that $Q' = \mathbf{0}$, and hence $P' \equiv Q'$.

- The cases $P' = n[S_1]$ and $P' = M.S_1$ are similar. As an example, we consider $P' = n[S_1]$. By definition, $[\![P']\!]_{fn(P')} = amb_n \circ (id_n \otimes [\![S_1]\!]_{fn(P')})$. Since $[\![P']\!]_{fn(P')}$ and $[\![Q']\!]_{fn(Q')}$ denote isomorphic graphs, they have the same interfaces, hence $fn(P') = fn(Q')$. Furthermore, there must be a bijective correspondence between the set of edges attached to the image of the input p of the graph $[\![P']\!]_{fn(P')}$, and the set of edges attached to the image of the input p of the graph $[\![Q']\!]_{fn(Q')}$. Analogously, there must be a bijective correspondence between the sets of edges attached to the image of the inputs a of both graphs. This means that $Q' = n[T_1]$, for some process T_1. We consider the graphical encoding for Q'. By definition, we have $[\![Q']\!]_{fn(Q')} = amb_n \circ (id_n \otimes [\![T_1]\!]_{fn(Q')})$. We know that $[\![P']\!]_{fn(P')} = [\![Q']\!]_{fn(Q')}$, hence it is obvious that $[\![S_1]\!]_{fn(P')} = [\![T_1]\!]_{fn(Q')}$. Since we also know that $fn(P') = fn(Q')$, then by using Lemma A.1, we deduce that $[\![S_1]\!]_{fn(S_1)} = [\![T_1]\!]_{fn(T_1)}$. Now, by applying the induction hypothesis, we have $S_1 \equiv T_1$, and therefore, thanks to the *Cong-Amb* rule, $S \equiv T$.

- Suppose that $P' = S_1 \mid S_2$. By definition $[\![P']\!]_{fn(P')} = [\![S_1]\!]_{fn(P')} \otimes [\![S_2]\!]_{fn(P')}$. Since $[\![P']\!]_{fn(P')}$ and $[\![Q']\!]_{fn(Q')}$ denote isomorphic graphs, they have the same interfaces, hence $fn(P') = fn(Q')$. Furthermore, there must be a bijective correspondence between the set of edges attached to the image of the input p of the graph $[\![P']\!]_{fn(P')}$, and the set of edges attached to the image of the input p of the graph $[\![Q']\!]_{fn(Q')}$. Analogously, there must be a bijective correspondence between the sets of edges attached to the image of the inputs a of both graphs. This means that P' and Q' have the same number of processes in parallel. So, since $[\![P']\!]_{fn(P')} = [\![Q']\!]_{fn(Q')}$, then there exist two processes T_1 and T_2, such that $T = T_1 \mid T_2$, and $[\![S_1]\!]_{fn(P')} = [\![T_1]\!]_{fn(Q')}$ and $[\![S_2]\!]_{fn(P')} = [\![T_2]\!]_{fn(Q')}$. Since we know that $fn(P') = fn(Q')$, then by Lemma A.1, we deduce that $[\![S_1]\!]_{fn(S_1)} = [\![T_1]\!]_{fn(T_1)}$ and $[\![S_2]\!]_{fn(S_2)} = [\![T_2]\!]_{fn(T_2)}$. Now, by applying the induction hypothesis, we have $S_1 \equiv T_1$ and $S_2 \equiv T_2$. So, thanks to the *Cong-Par* rule, $P' \equiv Q'$.

- Suppose that $P' = (\nu N)S$. By definition, we have

$$[\![P']\!]_{fn(P')} = \{\nu_{n_1} \otimes \{\{\nu_{n_2} \otimes \{\ldots \otimes \{(\nu_{n_i} \otimes [\![S]\!]_{fn(P') \cup \Gamma_{P'}}) \circ (\mathbf{0}_{n_i} \otimes id_{fn(P') \cup \Gamma_{P'} \setminus \{n_i\}})\} \circ \ldots\}\} \circ (\mathbf{0}_{n_2} \otimes id_{fn(P') \cup \{n_1\}})\}\} \circ (\mathbf{0}_{n_1} \otimes id_{fn(P')})$$

where $\Gamma_{P'} = \{n_1, \dots, n_i\} = N$. The value of the expression above is isomorphic to the value of

$$(\nu_{n_1} \otimes \dots \otimes \nu_{n_i} \otimes [\![S]\!]_{fn(P'))\cup\Gamma_{P'}}) \circ (\mathbf{0}_{\Gamma_{P'}} \otimes id_{fn(P')}) \ .$$

Since $[\![P']\!]_{fn(P')}$ and $[\![Q']\!]_{fn(Q')}$ denote isomorphic graphs, they have the same interfaces and hence $fn(P') = fn(Q')$.

Furthermore, there must be a bijective correspondence between the set of edges attached to the image of the input a of the graph $[\![P']\!]_{fn(P')}$, and the set of edges attached to the image of the input a of the graph $[\![Q']\!]_{fn(Q')}$. Analogously, there must be a bijective correspondence between the sets of nodes $\circ$ of both graphs. This means that P' and Q' have the same number of restricted names, hence, $Q' = (\nu m_1) \dots (\nu m_i)T$ for some process T.

By definition, we have

$$[\![Q']\!]_{fn(Q')} = \{\nu_{m_1}\otimes\{\{\nu_{m_2}\otimes\{\dots\otimes\{(\nu_{m_i}\otimes[\![T]\!]_{fn(Q')\cup\Gamma_{Q'}})\circ(\mathbf{0}_{m_i}\otimes id_{fn(Q')\cup\Gamma_{Q'}\setminus\{m_o\}})\}\circ \dots\}\} \circ (\mathbf{0}_{m_2} \otimes id_{fn(Q')\cup\{m_1\}})\}\} \circ (\mathbf{0}_{m_1} \otimes id_{fn(Q')})$$

where $\Gamma_{Q'} = \{m_1, \dots, m_i\}$. The value of the expression above is isomorphic to the value of

$$(\nu_{m_1} \otimes \dots \otimes \nu_{m_i} \otimes [\![T]\!]_{fn(Q'))\cup\Gamma_{Q'}}) \circ (\mathbf{0}_{\Gamma_{Q'}} \otimes id_{fn(Q')})$$

Since $[\![P']\!]_{fn(P')} = [\![Q']\!]_{fn(Q')}$ and P' and Q' have the same number of restricted names, then there exists a substitution σ such that $P'' = (\nu m_1) \dots (\nu m_i)S\sigma$ is α-equivalent to P' and $[\![S\sigma]\!]_{fn(P'')\cup\Gamma_{Q'}} = [\![T]\!]_{fn(Q')\cup\Gamma_{Q'}}$. We know that $fn(P'') = fn(Q')$, hence by using Lemma A.1 we deduce $[\![S\sigma]\!]_{fn(S\sigma)} = [\![T]\!]_{fn(T)}$. Now, we can apply the induction hypothesis and say that $S\sigma \equiv T$. So, thanks to the *Cong-Res* rule, we conclude that $P'' \equiv Q'$. Moreover, since $P'' \equiv P'$, and by Proposition A.3 we know that $P' \equiv P$ and $Q' \equiv Q$, then it is easy to conclude that $P \equiv Q$.

□

Now we show the completeness result by using Lemma A.2 and Lemma A.1.

Proposition A.4. Let P, Q be processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. If $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$, then $P \equiv Q$.

Proof. By Lemma A.1, we have $[\![P]\!]_\Gamma = [\![P]\!]_{fn(P)} \otimes free_\Gamma$, and analogously $[\![Q]\!]_\Gamma = [\![Q]\!]_{fn(Q)} \otimes free_\Gamma$. Since, by hypothesis, $[\![P]\!]_\Gamma = [\![Q]\!]_\Gamma$, then we have $[\![P]\!]_{fn(P)} = [\![Q]\!]_{fn(Q)}$. So, we can apply Lemma A.2 and conclude that $P \equiv Q$. □

A.2 From reduction relation → to graph rewriting

In this appendix we present the proofs of the two main results of Chapter 3. In particular, first we prove Theorem 3.2, which relates process reductions to graph rewrites. Then, we show the proof of the reverse direction, Theorem 3.3.

We begin by stating a useful lemma, saying that derivations are preserved under closure with respect to graph contexts. Intuitively, a graph context is a graph expression "with a hole", i.e., the single occurrence of a novel constant $-$.

Lemma A.3. *Let $\mathbb{G}$ be a graph with discrete interfaces, and let $C[-]$ be a graph context such that the graph expression $C[\mathbb{G}]$ is well-defined. If $\mathcal{R}_{amb}$ entails a direct derivation $\mathbb{G} \Longrightarrow \mathbb{H}$, then it also entails a direct derivation $C[\mathbb{G}] \Longrightarrow C[\mathbb{H}]$.*

A proof of a variant of the lemma above can be found in [32, Lemma B.3].

In order to prove the soundness and completeness results, we also need to introduce an extension of the encoding of processes into graphs, presented in Definition 3.16. So, in the following, we consider the process encoding as parametric with respect to the input interface. We denote by ${}^{a_1,p_1}[\![P]\!]_\Gamma$ the graph with interfaces $(\{a_1, p_1\}, \Gamma)$ that represents P, where Γ is a set of names such that $fn(P) \subseteq \Gamma$. Note that $[\![P]\!]_\Gamma = {}^{a,p}[\![P]\!]_\Gamma$.

Moreover, we use ${}^{a_1,p_1}amb_n^{a_2,p_2}$ to denote the constant graph with interfaces $(\{a_1, p_1\}, \{a_2, p_2, n\})$, corresponding to amb_n. Analogously, we use ${}^{a_1,p_1}act_n^{a_2,p_2}$ to denote the constant graph with interfaces $(\{a_1, p_1\}, \{a_2, p_2, n\})$, corresponding to act_n, while we write ${}^{a_1}go$ for the constant graph with interfaces $(\{a_1\}, \emptyset)$, corresponding to go. Finally, we use ${}^{a_1,p_1}id_{a,p}^{a_2,p_2}$ as shorthand for ${}^{a_1}id_a^{a_2} \otimes {}^{p_1}id_p^{p_2}$, where ${}^{a_1}id_a^{a_2}$ and ${}^{p_1}id_p^{p_2}$ respectively denote the constant graphs with interfaces $(\{a_1\}, \{a_2\})$ and $(\{p_1\}, \{p_2\})$, obviously corresponding to id_n.

Now, we prove that our encoding is sound with respect to $\rightarrow$.

Proof of Theorem 3.2. By induction on the depth of the derivation of $P \rightarrow Q$.

In order to prove the cases of *Red-In*, *Red-Out* and *Red-Open* rules, we follow the same pattern proposed in [32, Lemma C.1] to show the soundness of encoding with respect to the reduction semantics (even if for mobile ambients the case analysis is quite cumbersome). For each of these reduction rules, first, we choose a graph expression corresponding to the left-hand side of the respective rule in $\mathcal{R}_{amb}$. Then, we compute a graph expression corresponding to the application of the rule to the given graph expression. Finally, we show how the left-hand side occurs in the encoding $[\![P]\!]_\Gamma^{go}$, and we apply Lemma A.3.

- Assume that $P \rightarrow Q$ by *Red-In* rule. This means that $P = n[in\, m.R_2 \mid R_1] \mid m[R_3]$ and $Q = m[n[R_2 \mid R_1] \mid R_3]$.
  First of all, we consider a graph expression corresponding to the left-hand side of the rule p_{in} in $\mathcal{R}_{amb}$, and such that the source of the *go* edge occurs in the input interface, namely,

$$\mathbb{L}_{in} = {}^{a}go \otimes [\, {}^{a,p}amb_n^{a,p} \circ (id_n \otimes {}^{a,p}id_{a,p}^{a_1,p_1} \otimes {}^{a,p}in_m^{a_2,p_2})] \otimes {}^{a,p}amb_m^{a3,p_3} \quad .$$

  The application of the p_{in} rule with the identity match results in the value of

$$\begin{aligned}\mathbb{R}_{in} = {}^{a}go \otimes [\, & {}^{a,p}amb_m^{a,p} \circ (id_m \otimes {}^{a,p}amb_n^{a,p} \otimes {}^{a,p}id_{a,p}^{a_3,p_3}) \circ \\ & (id_m \otimes id_n \otimes {}^{a_3,p_3}id_{a,p}^{a_3,p_3} \otimes {}^{a,p}id_{a,p}^{a_1,p_1} \otimes {}^{a,p}id_{a,p}^{a_2,p_2})] \quad .\end{aligned}$$

  Now, we consider the graphical encoding for P. By definition, we have

$$\begin{aligned}[\![P]\!]_\Gamma^{go} = \{\{ & amb_n \circ \{id_n \otimes [in_m \circ (id_m \otimes [\![R_2]\!]_\Gamma)] \otimes [\![R_1]\!]_\Gamma\}\} \otimes \\ & [amb_m \circ (id_m \otimes [\![R_3]\!]_\Gamma)]\} \otimes go \quad .\end{aligned}$$

  The expression above can be rewritten to $\mathbb{L}_{in} \circ \mathbb{C}$, where

$$\mathbb{C} = id_n \otimes id_m \otimes {}^{a_2,p_2}[\![R_2]\!]_\Gamma \otimes {}^{a_1,p_1}[\![R_1]\!]_\Gamma \otimes {}^{a_3,p_3}[\![R_3]\!]_\Gamma \quad .$$

  Since, by applying the p_{in} rule, $\mathbb{L}_{in} \Longrightarrow \mathbb{R}_{in}$, then by Lemma A.3 we have $[\![P]\!]_\Gamma^{go} \Longrightarrow \mathbb{R}_{in} \circ \mathbb{C}$. Now, we have to show that the value of $\mathbb{R}_{in} \circ \mathbb{C}$ is isomorphic to the value of $[\![Q]\!]_\Gamma^{go}$. So, let us consider the graphical encoding for Q. By definition

$$\begin{aligned}[\![Q]\!]_\Gamma^{go} = \{ & amb_m \circ \{id_m \otimes [amb_n \circ (id_n \otimes [\![R_1]\!]_\Gamma \otimes [\![R_2]\!]_\Gamma)] \otimes \\ & [\![R_3]\!]_\Gamma\}\} \otimes go \quad .\end{aligned}$$

  It is easy to check that the value of the last expression is isomorphic to the value of $\mathbb{R}_{in} \circ \mathbb{C}$, hence the result holds.

- Suppose that $P \to Q$ has been obtained by applying *Red-Out* rule. It means that $P = m[n[out\ m.R_1 \mid R_2] \mid R_3]$ and $Q = n[R_1 \mid R_2] \mid m[R_3]$.
  Similarly as in the preceding case, first, we consider a graph expression corresponding to the left-hand side of the rule p_{out} in $\mathcal{R}_{amb}$, and such that the source of the *go* edge occurs in the input interface, namely,

  $$\begin{aligned}\mathbb{L}_{out} = {} & {}^{a}go \otimes [\, {}^{a,p}amb_m^{a,p} \circ (id_m \otimes {}^{a,p}amb_n^{a,p} \otimes {}^{a,p}id_{a,p}^{a_3,p_3}) \circ \\ & (id_m \otimes id_n \otimes {}^{a_3,p_3}id_{a,p}^{a_3,p_3} \otimes {}^{a,p}out_m^{a_1,p_1} \otimes {}^{a,p}id_{a,p}^{a_2,p_2})] \ .\end{aligned}$$

  The application of p_{out} with the identity match results in the value of

  $$\begin{aligned}\mathbb{R}_{out} = {} & {}^{a}go \otimes [\, {}^{a,p}amb_n^{a,p} \circ (id_n \otimes {}^{a,p}id_{a,p}^{a_1,p_1} \otimes {}^{a,p}id_{a,p}^{a_2,p_2})] \otimes \\ & {}^{a,p}amb_m^{a_3,p_3} \ .\end{aligned}$$

  Now, we consider the graphical encoding for P. By definition

  $$\begin{aligned}[\![P]\!]_\Gamma^{go} = {} & \{amb_m \circ \{id_m \otimes \{amb_n \circ \{id_n \otimes [out_m \circ (id_m \otimes [\![R_1]\!]_\Gamma)] \otimes \\ & [\![R_2]\!]_\Gamma\}\} \otimes [\![R_3]\!]_\Gamma\}\} \otimes go \ .\end{aligned}$$

  The expression above can be rewritten to $\mathbb{L}_{out} \circ \mathbb{C}$, where

  $$\mathbb{C} = id_n \otimes id_m \otimes {}^{a_1,p_1}[\![R_1]\!]_\Gamma \otimes {}^{a_2,p_2}[\![R_2]\!]_\Gamma \otimes {}^{a_3,p_3}[\![R_3]\!]_\Gamma \ .$$

  Since, by applying the p_{out} rule, $\mathbb{L}_{out} \Longrightarrow \mathbb{R}_{out}$, then by Lemma A.3 we have $[\![P]\!]_\Gamma^{go} \Longrightarrow \mathbb{R}_{out} \circ \mathbb{C}$. Now, we have to show that the value of $\mathbb{R}_{out} \circ \mathbb{C}$ is isomorphic to the value of $[\![Q]\!]_\Gamma^{go}$. So, let us consider the graphical encoding for Q. By definition

  $$\begin{aligned}[\![Q]\!]_\Gamma^{go} = {} & [amb_n \circ (id_n \otimes [\![R_1]\!]_\Gamma \otimes [\![R_2]\!]_\Gamma)] \otimes [amb_m \circ (id_m \otimes \\ & [\![R_3]\!]_\Gamma)] \otimes go \ .\end{aligned}$$

  It is easy to check that the value of the last expression is isomorphic to the value of $\mathbb{R}_{out} \circ \mathbb{C}$, hence the result holds.

- Assume that $P \to Q$ by *Red-Open* rule. It means that $P = open\ n.R_1 \mid n[R_2]$ and $Q = R_1 \mid R_2$.
  Consider a graph expression corresponding to the left-hand side of the rule p_{open} in $\mathcal{R}_{amb}$, and such that the source of the *go* edge occurs in the input interface, namely,

  $$\mathbb{L}_{open} = {}^{a}go \otimes {}^{a,p}open_n^{a_1,p_1} \otimes {}^{a,p}amb_n^{a_2,p_2} \ .$$

  The application of p_{open} with the identity match results in the value of

  $$\mathbb{R}_{open} = {}^{a}go \otimes {}^{a,p}id_{a,p}^{a_1,p_1} \otimes {}^{a,p}id_{a,p}^{a_2,p_2} \otimes free_n \ .$$

  Now, we consider the graphical encoding for P. By definition

  $$\begin{aligned}[\![P]\!]_\Gamma^{go} = {} & \{[open_n \circ (id_n \otimes [\![R_1]\!]_\Gamma)] \otimes [amb_n \circ (id_n \otimes [\![R_2]\!]_\Gamma)]\} \otimes \\ & go \ .\end{aligned}$$

  The expression above can be rewritten to $\mathbb{L}_{open} \circ \mathbb{C}$, where

  $$\mathbb{C} = id_n \otimes {}^{a_1,p_1}[\![R_1]\!]_\Gamma \otimes {}^{a_2,p_2}[\![R_2]\!]_\Gamma \ .$$

  Since, by applying the p_{open} rule $\mathbb{L}_{open} \Longrightarrow \mathbb{R}_{open}$, then by Lemma A.3, we have $[\![P]\!]_\Gamma^{go} \Longrightarrow \mathbb{R}_{open} \circ \mathbb{C}$.
  Now, we have to show that the value of $\mathbb{R}_{open} \circ \mathbb{C}$ is isomorphic to the value of $[\![Q]\!]_\Gamma^{go}$. So, let us consider the graphical encoding for Q. By definition

$$\llbracket Q \rrbracket^{go}_{\Gamma} = (\llbracket R_1 \rrbracket_{\Gamma} \otimes \llbracket R_2 \rrbracket_{\Gamma}) \otimes go \ \ .$$

It is easy to check that the value of the last expression is isomorphic to the value of $(\,{}^{a}go \otimes {}^{a,p}id^{a_1,p_1}_{a,p} \otimes {}^{a,p}id^{a_2,p_2}_{a,p}) \circ (\,{}^{a_1,p_1}\llbracket R_1 \rrbracket_{\Gamma} \otimes {}^{a_2,p_2}\llbracket R_2 \rrbracket_{\Gamma})$. Since $n \in \Gamma$, it is immediate to conclude that the value of $\llbracket Q \rrbracket^{go}_{\Gamma}$ is isomorphic to the value of $\mathbb{R}_{open} \circ \mathbb{C}$, hence the result holds.

- Suppose that $P \rightarrow Q$ has been obtained by applying *Red-Res* rule. It means that $P = (\nu n)P_1$, $Q = (\nu n)Q_1$ and $P_1 \rightarrow Q_1$. Consider the graph encodings $\llbracket P \rrbracket^{go}_{\Gamma}$ and $\llbracket Q \rrbracket^{go}_{\Gamma}$. By definition

$$\begin{aligned} \llbracket P \rrbracket^{go}_{\Gamma} &= [(\nu_m \otimes \llbracket P_1\{m/n\} \rrbracket_{\Gamma \cup \{m\}}) \circ (\mathbf{0}_m \otimes id_{\Gamma})] \otimes go \ \ , \\ \llbracket Q \rrbracket^{go}_{\Gamma} &= [(\nu_m \otimes \llbracket Q_1\{m/n\} \rrbracket_{\Gamma \cup \{m\}}) \circ (\mathbf{0}_m \otimes id_{\Gamma})] \otimes go \end{aligned}$$

for $m \notin \Gamma$. It is easy to see that the value of the first expression above is isomorphic to the value of the expression

$$(\nu_m \otimes \llbracket P_1\{m/n\} \rrbracket^{go}_{\Gamma \cup \{m\}}) \circ (\mathbf{0}_m \otimes id_{\Gamma}) \ \ .$$

Since $P_1 \rightarrow Q_1$, then $P_1\{m/n\} \rightarrow Q_1\{m/n\}$, and by induction hypothesis $\mathcal{R}_{amb}$ entails a direct derivation $\llbracket P_1\{m/n\} \rrbracket^{go}_{\Gamma \cup \{m\}} \Longrightarrow \mathbb{G}_1$, such that $\mathbb{G}_1 = \llbracket Q_1\{m/n\} \rrbracket^{go}_{\Gamma \cup \{m\}}$. So, we can apply Lemma A.3 and say that $\llbracket P \rrbracket^{go}_{\Gamma} \Longrightarrow (\nu_m \circ \llbracket Q_1\{m/n\} \rrbracket^{go}_{\Gamma \cup \{m\}}) \circ (\mathbf{0}_m \otimes id_{\Gamma})$. We conclude by observing that the value of $(\nu_m \otimes \llbracket Q_1\{m/n\} \rrbracket^{go}_{\Gamma \cup \{m\}}) \circ (\mathbf{0}_m \otimes id_{\Gamma})$ is isomorphic to the value of $\llbracket Q \rrbracket^{go}_{\Gamma}$.

- Assume that $P \rightarrow Q$ by *Red-Amb* rule. This means that $P = n[P_1]$, $Q = n[Q_1]$ and $P_1 \rightarrow Q_1$.
Now, we consider the graph encodings $\llbracket P \rrbracket^{go}_{\Gamma}$ and $\llbracket Q \rrbracket^{go}_{\Gamma}$. By definition

$$\begin{aligned} \llbracket P \rrbracket^{go}_{\Gamma} &= [amb_n \circ (id_n \otimes \llbracket P_1 \rrbracket_{\Gamma})] \otimes go \ \ , \\ \llbracket Q \rrbracket^{go}_{\Gamma} &= [amb_n \circ (id_n \otimes \llbracket Q_1 \rrbracket_{\Gamma})] \otimes go \ \ . \end{aligned}$$

It is easy to see that the value of the first expression above is isomorphic to the value of the following expression

$$amb_n \circ (id_n \otimes \llbracket P_1 \rrbracket^{go}_{\Gamma}) \ \ .$$

Since $P_1 \rightarrow Q_1$, by induction hypothesis $\mathcal{R}_{amb}$ entails a direct derivation $\llbracket P_1 \rrbracket^{go}_{\Gamma} \Longrightarrow \mathbb{G}_1$, such that $\mathbb{G}_1 = \llbracket Q_1 \rrbracket^{go}_{\Gamma}$. So, we can apply Lemma A.3 and say that $\llbracket P \rrbracket^{go}_{\Gamma} \Longrightarrow amb_n \circ (id_n \otimes \llbracket Q_1 \rrbracket^{go}_{\Gamma})$. We conclude by observing that the value of $amb_n \circ (id_n \otimes \llbracket Q_1 \rrbracket^{go}_{\Gamma})$ is isomorphic to the value of $\llbracket Q \rrbracket^{go}_{\Gamma}$.

- Suppose that $P \rightarrow Q$ has been obtained by applying *Red-Par* rule. It means that $P = P_1 \mid R$, $Q = Q_1 \mid R$ and $P_1 \rightarrow Q_1$.
Now, we consider the graph encodings $\llbracket P \rrbracket^{go}_{\Gamma}$ and $\llbracket Q \rrbracket^{go}_{\Gamma}$. By definition

$$\begin{aligned} \llbracket P \rrbracket^{go}_{\Gamma} &= (\llbracket P_1 \rrbracket_{\Gamma} \otimes \llbracket R \rrbracket_{\Gamma}) \otimes go \ \ , \\ \llbracket Q \rrbracket^{go}_{\Gamma} &= (\llbracket Q_1 \rrbracket_{\Gamma} \otimes \llbracket R \rrbracket_{\Gamma}) \otimes go \ \ . \end{aligned}$$

It is easy to see that the value of the first expression above is isomorphic to the value of the following expression:

$$\llbracket P_1 \rrbracket^{go}_{\Gamma} \otimes \llbracket R \rrbracket_{\Gamma} \ \ .$$

Since $P_1 \rightarrow Q_1$, by induction hypothesis $\mathcal{R}_{amb}$ entails a direct derivation $\llbracket P_1 \rrbracket^{go}_{\Gamma} \Longrightarrow \mathbb{G}_1$, such that $\mathbb{G}_1 = \llbracket Q_1 \rrbracket^{go}_{\Gamma}$. So, we conclude by observing that, by Lemma A.3, $\llbracket P \rrbracket^{go}_{\Gamma} \Longrightarrow \llbracket Q_1 \rrbracket^{go}_{\Gamma} \otimes \llbracket R \rrbracket_{\Gamma}$, and the value of $\llbracket Q_1 \rrbracket^{go}_{\Gamma} \otimes \llbracket R \rrbracket_{\Gamma}$ is isomorphic to the value of $\llbracket Q \rrbracket^{go}_{\Gamma}$

- Suppose that $P \rightarrow Q$ has been obtained by applying *Red-Cong* rule. This means that $P \equiv P_1$, $P_1 \rightarrow Q_1$ and $Q_1 \equiv Q$.
  Since $P \equiv P_1$, by Theorem 3.1, we have $[\![P]\!]_\Gamma = [\![P_1]\!]_\Gamma$. Analogously, since $Q_1 \equiv Q$, we have $[\![Q]\!]_\Gamma = [\![Q_1]\!]_\Gamma$. Moreover, by hypothesis we have $P_1 \rightarrow Q_1$, so we can apply the induction hypothesis and say that $\mathcal{R}_{amb}$ entails a direct derivation $[\![P_1]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$, such that $\mathbb{G} = [\![Q_1]\!]^{go}_\Gamma$. Now, we can notice that, since $[\![P]\!]_\Gamma = [\![P_1]\!]_\Gamma$, then $[\![P]\!]^{go}_\Gamma = [\![P_1]\!]^{go}_\Gamma$, and similarly since $[\![Q]\!]_\Gamma = [\![Q_1]\!]_\Gamma$, then $[\![Q]\!]^{go}_\Gamma = [\![Q_1]\!]^{go}_\Gamma$. So, it is immediate to conclude that $\mathcal{R}_{amb}$ entails a direct derivation $[\![P]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$, such that $\mathbb{G} = [\![Q]\!]^{go}_\Gamma$.

□

To prove the completeness of our encoding with respect to the reduction relation $\rightarrow$, we need to introduce two technical lemmas. The first states a property that characterises those graphs with interfaces in the image of the encoding.

Lemma A.4. *Let P be a process. If $\mathcal{R}_{amb}$ entails a derivation $[\![P]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$, then the graph with interfaces $\mathbb{G}$ satisfies the following property: the underlying graph is acyclic and only $\circ$ nodes may have more than one incoming tentacle. Moreover, the inputs (the node in the image of p and the node in the image of a) have no predecessors, and the outputs (the nodes in the image of Γ) have no successors.*

Sketch. The property clearly holds for all the graph constants used in the encoding. It is also easy to see that it is true for the graph expressions resulting from the encoding. In fact, the property is preserved by the parallel and sequential composition operators, because the interfaces are discrete. Moreover, since all the rules in $\mathcal{R}_{amb}$ also preserve the property, then the lemma holds. □

Now we introduce a simple result concerning the application of rules in $\mathcal{R}_{amb}$.

Lemma A.5. *Let $\mathbb{G}$ be a graph with discrete interfaces, and let $C[-]$ be a graph context, such that the graph expression $C[\mathbb{G}]$ is well-defined and the obvious morphism $\mathbb{G} \rightarrow C[\mathbb{G}]$ is mono. Moreover, let m be a match for the rule p in $\mathcal{R}_{amb}$, such that $p/m : C[\mathbb{G}] \Longrightarrow \mathbb{H}'$. If m covers the subgraph $\mathbb{G}$, then there exists a graph with interface $\mathbb{H}$, such that $\mathbb{G} \Longrightarrow \mathbb{H}$ and $\mathbb{H}' = C[\mathbb{H}]$.*

We now prove the completeness of our encoding with respect to the reduction semantics for processes with no restrictions on top.

Lemma A.6. *Let S be a process with no restriction operators on top and let Γ be a set of names, such that $fn(S) \subseteq \Gamma$. If $\mathcal{R}_{amb}$ entails a direct derivation $[\![S]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$, then there exists a process S' such that $S \rightarrow S'$ and $\mathbb{G} = [\![S']\!]^{go}_\Gamma$.*

Proof. The proof proceeds by induction on the structure of S which, with no loss of generality, could be considered in normal form

- Assume $S = \mathbf{0}$ or $S = M.S_1$. In both cases the proof is trivial, because there is no derivation from $[\![S]\!]^{go}_\Gamma$.
- Assume $S = n[S_1]$. By definition, we have $[\![n[S_1]]\!]^{go}_\Gamma = [amb_n \circ (id_n \otimes [\![S_1]\!]_\Gamma)] \otimes go$. We notice that this last expression can be rewritten to $amb_n \circ (id_n \otimes [\![S_1]\!]^{go}_\Gamma)$. Moreover, note that the derivation $[\![S]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$ via production p and match m' could have been obtained in two ways:
  1. the match covers only the graph $[\![S_1]\!]^{go}_\Gamma$;
  2. the match covers both the graphs $[\![S_1]\!]^{go}_\Gamma$ and amb_n.

  1. Suppose that the match m' covers only the graph $[\![S_1]\!]^{go}_\Gamma$. So, by Lemma A.5, there exists a graph with interfaces $\mathbb{G}_1$ such that $[\![S_1]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}_1$ and $\mathbb{G} = amb_n \circ (id_n \otimes \mathbb{G}_1)$. Since $[\![S_1]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}_1$, we can apply the induction hypothesis and say that there exists a process S_1', such that $S_1 \rightarrow S_1'$ and $\mathbb{G}_1 = [\![S_1']\!]^{go}_\Gamma$.
     Let us recall that we have to prove that there exists a process S' such that $S \rightarrow S'$ and $\mathbb{G} = [\![S']\!]^{go}_\Gamma$. We take $S' = n[S_1']$. Since $S_1 \rightarrow S_1'$, then by applying the *Red-Amb* rule we have

$S \rightarrow S'$. Moreover, we know that $\mathbb{G}_1 = [\![S'_1]\!]^{go}_\Gamma$, hence we have $\mathbb{G} = amb_n \circ (id_n \otimes [\![S'_1]\!]^{go}_\Gamma)$. We conclude by observing that the value of $amb_n \circ (id_n \otimes [\![S'_1]\!]^{go}_\Gamma)$ is isomorphic to the value of $[\![n[S'_1]]\!]^{go}_\Gamma$.

2. Assume that the match m' covers both the graphs $[\![S_1]\!]^{go}_\Gamma$ and amb_n. In this case, the rewriting step could have been obtained only by applying the p_{out} rule. The graph $[\![S]\!]^{go}_\Gamma$ has interfaces $(\{a,p\},\Gamma)$ and exactly one occurrence of a go edge, which is outgoing from the image of the input a. Moreover, since $[\![S]\!]^{go}_\Gamma$ satisfies the property stated in Lemma A.4, any match m' for the rule p_{out} has to be injective, at most coalescing the $\circ$ nodes corresponding to the names m and n of rule p_{out} in Figure 3.16. So, the graphical encoding for S has to have the following shape

$$[\![S]\!]^{go}_\Gamma = \mathbb{L}_{out} \circ \mathbb{C}$$

where $\mathbb{C} = id_n \otimes id_m \otimes {}^{a_1,p_1}[\![T_1]\!]_\Gamma \otimes {}^{a_2,p_2}[\![T_2]\!]_\Gamma \otimes {}^{a_3,p_3}[\![T_3]\!]_\Gamma$, for T_1, T_2 and T_3 processes and n and m ambient names, while $\mathbb{L}_{out}$ is the expression corresponding to the left-hand side of the p_{out} rule, shown in the proof of Theorem 3.2. So, we have that $\mathbb{G} = \mathbb{R}_{out} \circ \mathbb{C}$.
Now, we notice that the value of $[\![S]\!]^{go}_\Gamma$ is isomorphic to the value of $[\![T]\!]^{go}_\Gamma$, for $T = m[n[out\ m.T_1 \mid T_2] \mid T_3]$. So, by Theorem 3.1 $S \equiv T$.
Let us recall that we have to prove that there exists a process S' such that $S \rightarrow S'$ and $\mathbb{R}_{out} \circ \mathbb{C} = [\![S']\!]^{go}_\Gamma$. We take $S' = n[T_1 \mid T_2] \mid m[T_3]$. Since $T \rightarrow S'$ by applying ***Red-Out*** rule, and $S \equiv T$, we have $S \rightarrow S'$ by ***Red-Cong*** rule.
Now, we consider the graphical encoding for S'. By definition, we have

$$[\![S']\!]^{go}_\Gamma = [amb_n \circ (id_n \otimes [\![T_1]\!]_\Gamma \otimes [\![T_2]\!]_\Gamma)] \otimes [amb_m \circ (id_m \otimes [\![T_3]\!]_\Gamma)] \otimes go\ .$$

It is easy to check that the value of the last expression is isomorphic to the value of $\mathbb{R}_{out} \circ \mathbb{C}$, hence the result holds.

- Assume $S = S_1 \mid S_2$. By definition, we have $[\![S_1 \mid S_2]\!]^{go}_\Gamma = [[\![S_1]\!]_\Gamma \otimes [\![S_2]\!]_\Gamma] \otimes go$. We notice that this last expression can be rewritten to $[\![S_1]\!]^{go}_\Gamma \otimes [\![S_2]\!]_\Gamma$ or to $[\![S_1]\!]_\Gamma \otimes [\![S_2]\!]^{go}_\Gamma$. Moreover, we note that the derivation $[\![S]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$ via production p and match m' could have been obtained in three ways

1. the match covers only one of graphs $[\![S_1]\!]^{go}_\Gamma$ and $[\![S_2]\!]^{go}_\Gamma$;
2. the match covers both the graphs $[\![S_1]\!]^{go}_\Gamma$ and $[\![S_2]\!]_\Gamma$ and p_{in} rule has been applied;
3. the match covers both the graphs $[\![S_1]\!]^{go}_\Gamma$ and $[\![S_2]\!]_\Gamma$ and p_{open} rule has been applied.

1. Suppose that the match m' covers only the graph $[\![S_1]\!]^{go}_\Gamma$. So, by Lemma A.5, there exists a graph with interfaces $\mathbb{G}_1$, such that $[\![S_1]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}_1$ and $\mathbb{G} = \mathbb{G}_1 \otimes [\![S_2]\!]_\Gamma$. Since $[\![S_1]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}_1$, we can apply the induction hypothesis and say that there exists a process S'_1, such that $S_1 \rightarrow S'_1$ and $\mathbb{G}_1 = [\![S'_1]\!]^{go}_\Gamma$.
Let us recall that we have to prove that there exists a process S', such that $S \rightarrow S'$ and $\mathbb{G} = [\![S']\!]^{go}_\Gamma$. We take $S' = S'_1 \mid S_2$. Since $S_1 \rightarrow S'_1$, then by applying ***Red-Par*** rule we have $S \rightarrow S'$. Moreover, we know that $\mathbb{G}_1 = [\![S'_1]\!]^{go}_\Gamma$, hence we have $\mathbb{G} = [\![S'_1]\!]^{go}_\Gamma \otimes [\![S_2]\!]_\Gamma$. We conclude by observing that the value of $[\![S'_1]\!]^{go}_\Gamma \otimes [\![S_2]\!]_\Gamma$ is isomorphic to the value of $[\![S'_1 \mid S_2]\!]^{go}_\Gamma$.
2. Assume that the match m' covers both graphs $[\![S_1]\!]^{go}_\Gamma$ and $[\![S_2]\!]_\Gamma$ and the p_{in} rule has been applied. It means that the gluing condition is satisfied, hence any match m' for the rule p_{in} can not identify the two amb edges of the left-hand side of the p_{in} rule in Figure 3.16. Moreover, the graph $[\![S]\!]^{go}_\Gamma$ satisfies the property stated in Lemma A.4. Hence, any match m' for the rule p_{in} has to be injective, at most coalescing the $\circ$ nodes corresponding to the names m and n of the p_{in} rule in Figure 3.16. We note that the graph $[\![S]\!]^{go}_\Gamma$ has interfaces $(\{a,p\},\Gamma)$ and exactly one occurrence of a go edge, which is outgoing from the image of the input a. From this follows that the graphical encoding for S has to have the following shape

$$[\![S]\!]^{go}_\Gamma = (\mathbb{L}_{in} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$$

where $\mathbb{C} = id_n \otimes id_m \otimes {}^{a_1,p_1}[\![T_1]\!]_\Gamma \otimes {}^{a_2,p_2}[\![T_2]\!]_\Gamma \otimes {}^{a_3,p_3}[\![T_3]\!]_\Gamma$, for T_1, T_2, T_3 and S_3 processes, and n and m ambient names, while $\mathbb{L}_{in}$ is the expression corresponding to the left-hand side of the p_{in} rule, shown in the proof of Theorem 3.2. So, we have that $\mathbb{G} = (\mathbb{R}_{in} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$.
Now, we note that the value of $[\![S]\!]^{go}_\Gamma$ is isomorphic to that of $[\![T]\!]^{go}_\Gamma$, for $T = n[in\ m.T_2 \mid T_1] \mid m[T_3] \mid S_3$. So $S \equiv T$ by Theorem 3.1.
Let us recall that we have to prove that there exists a process S', such that $S \to S'$ and $(\mathbb{R}_{in} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma = [\![S']\!]^{go}_\Gamma$. We take $S' = m[n[T_1 \mid T_2] \mid T_3] \mid S_3$. Since $T \to S'$ by applying *Red-In* and *Red-Par* rules, and $S \equiv T$, we have $S \to S'$ by *Red-Cong* rule.
Now, we consider the graphical encoding for S'. By definition, we have

$$[\![S']\!]^{go}_\Gamma = \{amb_m \circ \{id_m \otimes [amb_n \circ (id_n \otimes [\![T_1]\!]_\Gamma \otimes [\![T_2]\!]_\Gamma)] \otimes [\![T_3]\!]_\Gamma\}\} \otimes [\![S_3]\!]_\Gamma \otimes go \ .$$

It is easy to check that the value of the last expression is isomorphic to the value of $(\mathbb{R}_{in} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$, hence the result holds.

3. Suppose that the match m' covers both graphs $[\![S_1]\!]^{go}_\Gamma$ and $[\![S_2]\!]_\Gamma$ and the p_{open} rule has been applied. The graph $[\![S]\!]^{go}_\Gamma$ has interfaces $(\{a, p\}, \Gamma)$ and exactly one occurrence of a go edge, which is outgoing from the image of the input a. Moreover, since $[\![S]\!]^{go}_\Gamma$ satisfies the property stated in Lemma A.4, any match m' for the rule p_{out} has to be injective. So, we have that the graphical encoding for S has to have the following shape

$$[\![S]\!]^{go}_\Gamma = (\mathbb{L}_{open} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$$

where $\mathbb{C} = id_n \otimes {}^{a_1,p_1}[\![T_1]\!]_\Gamma \otimes {}^{a_2,p_2}[\![T_2]\!]_\Gamma$, for T_1, T_2 and S_3 processes and n an ambient name, while $\mathbb{L}_{open}$ is the expression corresponding to the left-hand side of the p_{open} rule, shown in the proof of Theorem 3.2. So, we have that $\mathbb{G} = (\mathbb{R}_{open} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$.
Now, we notice that the value of $[\![S]\!]^{go}_\Gamma$ is isomorphic to the value of $[\![T]\!]^{go}_\Gamma$, for $T = open\ n.T_1 \mid n[T_2] \mid S_3$. Hence $S \equiv T$ by Theorem 3.1.
Let us recall that we have to prove that there exists a process S', such that $S \to S'$ and $\mathbb{G} = [\![S']\!]^{go}_\Gamma$. We take $S' = T_1 \mid T_2 \mid S_3$. Since $T \to S'$ by applying *Red-Open* and *Red-Par* rules, and $S \equiv T$, we have $S \to S'$ by *Red-Cong* rule.
Now, we consider the graphical encoding for S'. By definition, we have

$$[\![S']\!]^{go}_\Gamma = [\![T_1]\!]_\Gamma \otimes [\![T_2]\!]_\Gamma \otimes [\![S_3]\!]_\Gamma \otimes go \ .$$

It is easy to check that the value of the last expression is isomorphic to the value of $(\mathbb{R}_{open} \circ \mathbb{C}) \otimes [\![S_3]\!]_\Gamma$, hence the result holds.

□

Now, we can at last show the proof of Theorem 3.3.

Proof of Theorem 3.3. Let $P' = (\nu n_1) \ldots (\nu n_i)S$ be the normal form of P, such that $\forall j : n_j \notin \Gamma$. If $i = 0$, that is, P' is a process without restrictions as top operators, the result holds thanks to Lemma A.6. If $i > 0$, by definition, we have

$$[\![P]\!]_\Gamma = \{\nu_{n_1} \otimes \{\{\nu_{n_2} \otimes \{\ldots \otimes \{(\nu_{n_i} \otimes [\![S]\!]_{\Gamma \cup \Gamma_{P'}}) \circ (\mathbf{0}_{n_i} \otimes id_{\Gamma \cup \Gamma_{P'} \setminus \{n_i\}})\} \circ \ldots\}\} \circ (\mathbf{0}_{n_2} \otimes id_{\Gamma \cup \{n_1\}})\}\} \circ (\mathbf{0}_{n_1} \otimes id_\Gamma)$$

where $\Gamma_{P'} = \{n_1, \ldots, n_i\}$.
The value of the expression above is isomorphic to the value of the following

$$(\nu_{n_1} \otimes \ldots \otimes \nu_{n_i} \otimes [\![S]\!]_{\Gamma \cup \Gamma_{P'}}) \circ (\mathbf{0}_{\Gamma_{P'}} \otimes id_\Gamma) \ .$$

Note that, by Proposition A.3, we have $P \equiv P'$, hence $[\![P]\!]^{go}_\Gamma = [\![P']\!]^{go}_\Gamma$. Since, by hypothesis, $[\![P]\!]^{go}_\Gamma \Longrightarrow \mathbb{G}$, and any match covers only $[\![S]\!]_{\Gamma \cup \Gamma_{P'}}$, by Lemma A.5, there exists a graph $\mathbb{G}_1$, such that $[\![S]\!]_{\Gamma \cup \Gamma_{P'}} \Longrightarrow \mathbb{G}_1$ and $\mathbb{G} = \mathbb{G}_1 \circ (\mathbf{0}_{\Gamma_{P'}} \otimes id_\Gamma)$. Since $[\![S]\!]_{\Gamma \cup \Gamma_{P'}} \Longrightarrow \mathbb{G}_1$, we can apply Lemma A.6 and say that there

exists a process S', such that $S \rightarrow S'$ and $\mathbb{G}_1 = [\![S']\!]^{go}_{\Gamma \cup \Gamma_{P'}}$.
Let us recall that we have to prove that there exists a process Q, such that $P \rightarrow Q$ and $\mathbb{G} = [\![Q]\!]^{go}_{\Gamma}$. We take $Q = (\nu n_1) \ldots (\nu n_i) S'$. Since $S \rightarrow S'$, then by applying *Red-Res* rule, we have $P' \rightarrow Q$. We also know that $P \equiv P'$, so by *Red-Cong* rule we have $P \rightarrow Q$. Moreover, since $\mathbb{G}_1 = [\![S']\!]^{go}_{\Gamma \cup \Gamma_{P'}}$, then $\mathbb{G} = (\nu_{n_1} \otimes \ldots \otimes \nu_{n_i} \otimes [\![S']\!]^{go}_{\Gamma \cup \Gamma_{P'}}) \circ (\mathbf{0}_{\Gamma_{P'}} \otimes id_{\Gamma})$. We conclude by observing that the value of the last expression is isomorphic to the value of $[\![Q]\!]^{go}_{\Gamma}$. $\square$

A.3 Collecting useless restrictions

In this appendix we turn our attention to the proofs of Proposition 3.3 and Theorem 3.4, which formalise respectively the relation between the structural congruence $\equiv'$ and the encoding introduced in Definition 3.16, and the relation between process reductions according to $\rightarrow'$ and graph rewrites.

We begin by a lemma showing that the encoding of reductions are closed with respect to the removal of restrictions.

Lemma A.7. *Let P, Q be processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. If $P \rightarrow Q$ then $nf([\![P]\!]_{\Gamma}) \Longrightarrow nf([\![Q]\!]_{\Gamma})$.*

The proof exploits an obvious extension of Proposition 3.2, which is applied to finite sequences of derivations.

Lemma A.8. *Let P, Q be processes and let Γ be a set of names, such that $fn(P) \cup fn(Q) \subseteq \Gamma$. If $P \equiv' Q$ then $nf([\![P]\!]_{\Gamma}) = nf([\![Q]\!]_{\Gamma})$.*

The proof of the result above, the soundness of Proposition 3.3, is straightforward: either $P \equiv Q$, or at least once the law $(\nu n)\mathbf{0} = \mathbf{0}$ has been applied. Since $p_{\nu}/id : [\![(\nu n)\mathbf{0}]\!]_{\Gamma} \Longrightarrow [\![\mathbf{0}]\!]_{\Gamma}$, the proof goes by induction on the derivation, proving that rewrites with p_{ν} are closed with respect to context application.

The completeness amounts to prove the following proposition.

Proposition A.5. If $p_{\nu}/m : [\![P]\!]_{\Gamma} \Longrightarrow G$ then there exists Q such that $P \equiv' Q$ and $[\![Q]\!]_{\Gamma} = G$.

The proof proceeds by induction on the structure of P which, with no loss of generality, could be considered in normal form. The only interesting case is when $P = ((\nu N_r)S) \mid ((\nu N_g)\mathbf{0})$. It is easy to check that if $p_{\nu}/m : [\![P]\!]_{\Gamma} \Longrightarrow G$, then G is $[\![P]\!]_{\Gamma}$ without the graphical restriction operator for a name n_j in N_g. It is the graphical encoding of $Q = ((\nu N_r)S) \mid ((\nu(N_g \setminus \{n_j\})\mathbf{0})$, and obviously $P \equiv' Q$.

Proof of Theorem 3.4. Let us assume that $P \rightarrow' Q$. Then, there exists R such that $P \rightarrow R$ and $R \equiv' Q$ by Proposition 3.1. This implies that $nf([\![P]\!]_{\Gamma}) \Longrightarrow nf([\![R]\!]_{\Gamma})$ and $nf([\![P]\!]_{\Gamma}) \Longrightarrow nf([\![R]\!]_{\Gamma})$ by the lemmas above.

Vice versa, let us assume that $nf([\![P]\!]_{\Gamma}) \Longrightarrow G$. Then, there exists H such that $[\![P]\!]_{\Gamma} \Longrightarrow H$ and H reaches G by a sequence of derivations applying the rule p_{ν}, so $nf(H) = nf(G)$. Hence $P \rightarrow Q$ and $[\![Q]\!]_{\Gamma} = H$, so $nf([\![P]\!]_{\Gamma}) \Longrightarrow nf([\![Q]\!]_{\Gamma})$ and $nf([\![Q]\!]_{\Gamma}) = nf(G)$. $\square$

Appendix B

Proofs of Chapter 4

B.1 Equivalence between the LTS $\mathcal{D}_\mathcal{J}$ and the LTS $\mathcal{S}_\mathcal{J}$

This section discusses the equivalence between the LTS $\mathcal{D}_\mathcal{J}$, presented in Section 4.6, and the LTS $\mathcal{S}_\mathcal{J}$, introduced in Section 4.7. In particular, we provide a proof of Theorem 4.2, and to this end, in the following we introduce two useful propositions.

Proposition B.1. Let P be a pure process. If $P \xrightarrow{C[-]_\epsilon}_\mathcal{D} Q_\epsilon$ then there exists a well-formed process Q'_ϵ such that $P \xrightarrow{C[-]_\epsilon}_\mathcal{S} Q'_\epsilon$ and for each substitution σ, $Q_\epsilon\sigma \equiv Q'_\epsilon\sigma$.

Sketch. We begin by observing that the rules in Figure 4.17 and the rules in the first two rows of Figure 4.19 exactly derive the same transition relation of the reduction relation of mobile ambients. So for them the proposition trivially holds.

The proof is by cases on the rules to obtain $P \xrightarrow{C[-]_\epsilon}_\mathcal{D} Q_\epsilon$.

For the rules in Figure 4.18 we show as an example the case of the IN rule.

Assume that $P \xrightarrow{C_\epsilon[-]}_\mathcal{D} Q_\epsilon$ by IN rule. It means that $P \equiv (\nu A)(in\ m.P_1|P_2)$, $m \notin A$, $Q_\epsilon = (\nu A)(m[x[P_1|P_2|X_1]|X_2])$ and $C_\epsilon[-] = x[-|X_1]|m[X_2]$.

We can note that, by applying IN rule, $in\ m.P_1 \xrightarrow{x[-|X_1]|m[X_2]}_\mathcal{S} m[x[P_1|X_1]|X_2]$. So, we can apply INPAR rule and obtain $in\ m.P_1|P_2 \xrightarrow{x[-|X_1]|m[X_2]}_\mathcal{S} m[x[P_1|P_2|X_1]|X_2]$. Since we also know $m \notin A$, thanks to the rule INRES, we can conclude $(\nu A)(in\ m.P_1|P_2) \xrightarrow{x[-|X_1]|m[X_2]}_\mathcal{S} Q_\epsilon$, therefore $P \xrightarrow{x[-|X_1]|m[X_2]}_\mathcal{S} Q_\epsilon$ and trivially, for each substitution σ, $Q_\epsilon\sigma \equiv Q_\epsilon\sigma$. □

Proposition B.2. Let P be a pure process. If $P \xrightarrow{C[-]_\epsilon}_\mathcal{S} Q_\epsilon$ then there exists a well-formed process Q'_ϵ such that $P \xrightarrow{C[-]_\epsilon}_\mathcal{D} Q'_\epsilon$ and for each substitution σ, $Q_\epsilon\sigma \equiv Q'_\epsilon\sigma$.

Sketch. We proceed by induction on the depth of the derivation of $P \xrightarrow{C[-]_\epsilon}_\mathcal{S} Q_\epsilon$.

As in the proof above, for the rules in the first two rows of Figure 4.19 the proposition trivially holds. Instead, for the remaining rules of the same figure, we show as an example the cases for IN, INPAR and CONG rules.

- Assume that $P \xrightarrow{C_\epsilon[-]}_\mathcal{S} Q_\epsilon$ by IN rule of Figure 4.19. It means that $P = in\ m.P_1$, $Q_\epsilon = m[x[P_1|X_1]|X_2]$ and $C_\epsilon[-] = x[-|X_1]|m[X_2]$. It is easy to check that $P \xrightarrow{x[-|X_1]|m[X_2]}_\mathcal{D} Q_\epsilon$ by IN rule of Figure 4.18, so the proposition trivially holds.

- Assume that $P \xrightarrow{C_\epsilon[-]}_\mathcal{S} Q_\epsilon$ by INPAR rule. This means that $P = P'|R'$, $C_\epsilon[-] = x[-|X_1]|m[X_2]$, $P' \xrightarrow{C_\epsilon[-]}_\mathcal{S} Q'_\epsilon$ and $Q_\epsilon = Q''_\epsilon\{^{R'|X_1}/_{X_1}\}$.

By induction hypothesis, we have $P' \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{D}} Q''_\epsilon$. This means that $P' \equiv (\nu A)(in\ m.P_1|P_2)$, $m \notin A$ and $Q''_\epsilon = (\nu A)(m[x[P_1|P_2|X_1]|X_2])$.

Note that $P'|R' \equiv (\nu A)(in\ m.P_1|P_2)|R'$ and $(\nu A)(in\ m.P_1|P_2)|R' \equiv (\nu A')(in\ m.P'_1|P'_2|R')$, by considering $(\nu A)(in\ m.P_1|P_2)$ α-equivalent to $(\nu A')(in\ m.P'_1|P'_2)$ and $A' \cap fn(R') = \emptyset$.

So, thanks to IN rule, $P'|R' \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{D}} Q'_\epsilon$, where $Q'_\epsilon = (\nu A')(m[x[P'_1|P'_2|R'|X_1]|X_2])$ and it is easy to check that for each substitution σ, $Q_\epsilon\sigma \equiv Q'_\epsilon\sigma$.

- Assume that $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ by CONG rule. This means that $P \equiv P'$, $P' \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$. By induction hypothesis, we have $P' \xrightarrow{C_\epsilon[-]}_{\mathcal{D}} Q_\epsilon$, hence also $P \xrightarrow{C_\epsilon[-]}_{\mathcal{D}} Q_\epsilon$ and so the proposition trivially holds.

□

Theorem 4.2 trivially follows from the two propositions above and from Definitions 4.6 and 4.7.

B.2 Correspondence between the LTS $\mathcal{S}_\mathcal{I}$ and the LTS $\mathcal{CA}$

This section shows the proofs of Propositions 4.3 and 4.4 used to formally prove the correspondence between our LTS $\mathcal{S}_I$, defined on pure processes of mobile ambients, and the LTS $\mathcal{CA}$ for mobile ambients proposed by Rathke and Sobociński in [60].

First of all, we introduce the proof of Proposition 4.3, needed to prove the first statement of Theorem 4.3.

Proof sketch of Proposition 4.3. The proof is by cases on the rules to obtain $P \xrightarrow{\alpha\downarrow\vec{M}^\alpha}_{\mathcal{CA}} Q$. We only show the proof for some rules, because the other cases are analogous.

- Assume that $P \xrightarrow{\alpha\downarrow\vec{M}^\alpha}_{\mathcal{CA}} Q$ by $C\lambda$ rule. It means that $P \xrightarrow{\alpha}_{\mathcal{C}} A$, $A \xrightarrow{\vec{M}^\alpha\downarrow}_{\mathcal{A}} Q$ and $\alpha \notin \{[\overline{in\,m}], \overline{open\,n}, \tau\}$. Now we proceed by cases on the rules to obtain $P \xrightarrow{\alpha}_{\mathcal{C}} A$ with $\alpha \notin \{[\overline{in\,m}], \overline{open\,n}, \tau\}$. As an example, we show the cases of the IN and $||$IN rules.

  Assume that $P \xrightarrow{\alpha}_{\mathcal{C}} A$ by IN rule. It means that $P = in\ m.P_1$, $A = \lambda XxY.m[x[P_1|X]|Y]$ and $\alpha = in\ m$. We assume that $\vec{M}^\alpha = R, n, S$, for R, S processes and n ambient name, therefore we have $Q = m[n[P_1|R]|S]$.

  We have to show that there exists Q_ϵ, such that $P \xrightarrow{C^{in\ m}_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ and $Q \equiv Q_\epsilon\sigma^\alpha_M$, with $\sigma^\alpha_M = \{^R/_{X_1}, ^n/_x, ^S/_{X_2}\}$. We take $Q_\epsilon = m[x[P_1|X_1]|X_2]$. It is easy to check that $in\ m.P_1 \xrightarrow{C^{in\ m}_\epsilon[-]}_{\mathcal{S}} m[x[P_1|X_1]|X_2]$ by IN rule. Moreover, we have $m[x[P_1|X_1]|X_2]\sigma^\alpha_M = m[n[P_1|R]|S]$.

  Assume that $P \xrightarrow{\alpha}_{\mathcal{C}} A$ by $||$IN rule. It means that $\alpha = in\ m$, $P = P_1|P_2$, $P_1 \xrightarrow{in\ m}_{\mathcal{C}} A'$, and $A = \lambda X.A'(P_2|X)$. We assume that $\vec{M}^\alpha = R, n, S$, for R, S processes and n ambient name, therefore we have $Q = A'(P_2|R, n, S)$. Let us consider $\vec{M'}^\alpha = P_2|R, n, S$. Since $A \xrightarrow{\vec{M}^\alpha\downarrow}_{\mathcal{A}} Q$, then $A' \xrightarrow{\vec{M'}^\alpha\downarrow}_{\mathcal{A}} Q$. Therefore, we have $P_1 \xrightarrow{in\ m\downarrow\vec{M'}^\alpha}_{\mathcal{CA}} Q$. By induction hypothesis, there exists Q'_ϵ such that $P_1 \xrightarrow{C^{in\ m}_\epsilon[-]}_{\mathcal{S}} Q'_\epsilon$ and $Q \equiv Q'_\epsilon\sigma^\alpha_{M'}$, where $\sigma^\alpha_{M'} = \{^{P_2|R}/_{X_1}, ^n/_x, ^S/_{X_2}\}$. We have to show that there exists Q_ϵ, such that $P \xrightarrow{C^{in\ m}_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ and $Q \equiv Q_\epsilon\sigma^\alpha_M$, with $\sigma^\alpha_M = \{^R/_{X_1}, ^n/_x, ^S/_{X_2}\}$. We take $Q_\epsilon = Q'_\epsilon\{^{P_2|X_1}/_{X_1}\}$. It is easy to check that $P_1|P_2 \xrightarrow{C^{in\ m}_\epsilon[-]}_{\mathcal{S}} Q'_\epsilon\{^{P_2|X_1}/_{X_1}\}$ by INPAR rule. Moreover, it is obvious that $Q'_\epsilon\{^{P_2|X_1}/_{X_1}\}\{^R/_{X_1}, ^n/_x, ^S/_{X_2}\} = Q'_\epsilon\{^{P_2|R}/_{X_1}, ^n/_x, ^S/_{X_2}\} \equiv Q$.

- Assume that $P \xrightarrow{\alpha\downarrow\vec{M}^\alpha}_{\mathcal{CA}} Q$ by COINλ rule. This means that $\alpha = [\overline{in\,m}]$, $P \xrightarrow{[\overline{in\,m}]}_{\mathcal{C}} A$, $\vec{M}^\alpha = R, S, n$ and $A(\lambda XYZx.m[x[Y|Z]|X]) \xrightarrow{R,S,n\downarrow}_{\mathcal{A}} Q$. Now we proceed by cases on the rules to obtain $P \xrightarrow{[\overline{in\,m}]}_{\mathcal{C}} A$. As an example, we show the case of the COIN rule.

  Assume that $P \xrightarrow{[\overline{in\,m}]}_{\mathcal{C}} A$ by COIN rule. It means that $P = m[P_1]$ and $A = \lambda Z.Z(P_1)$, and hence we have $Q = m[n[R|S]|P_1]$.

We have to show that there exists Q_ϵ, such that $P \xrightarrow{C_\epsilon^{[in\ m]}[-]}_{\mathcal{S}} Q_\epsilon$ and $Q \equiv Q_\epsilon \sigma_M^\alpha$, with $\sigma_M^\alpha = \{^R/_{X_1}, ^S/_{X_2}, ^n/_x\}$. We take $Q_\epsilon = m[x[X_1|X_2]|P_1]$. It is easy to check that $m[P_1] \xrightarrow{C_\epsilon^{[in\ m]}[-]}_{\mathcal{S}} m[x[X_1|X_2]|P_1]$ by COIN rule. and $m[x[X_1|X_2]|P_1]\{^R/_{X_1}, ^S/_{X_2}, ^n/_x\} = m[n[R|S]|P_1]$.

□

Now we show the proof of Proposition 4.4, needed to prove the second statement of Theorem 4.3.

Proof sketch of Proposition 4.4. The proof proceeds by induction on the depth of the derivation $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$. We only show some cases, because the other ones are analogous.

- Assume that $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ by IN rule. This means that $\alpha = in\ m$, $P = in\ m.P_1$, $C_\epsilon^{in\ m}[-] = x[-|X_1]|m[X_2]$ and $Q_\epsilon = m[x[P_1|X_1]|X_2]$. Moreover, the substitution σ has the following shape $\{^{P_2}/_{X_1}, ^n/_x, ^{P_3}/_{X_2}\}$, for some ambient name n and some processes P_1 and P_2. Therefore, we have $Q \equiv Q_\epsilon\sigma = m[x[P_1|X_1]|X_2]\{^{P_2}/_{X_1}, ^n/_x, ^{P_3}/_{X_2}\} = m[n[P_1|P_2]|P_3]$. We have to show that $P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q$, where $\vec{M}_\sigma^\alpha = P_2, n, P_3$. It is easy to check that $P \xrightarrow{in\ m}_{\mathcal{C}} \lambda X x Y.m[x[P_1|X]|Y]$ tanks to IN rule in Figure 6 of [60]. Moreover, we can apply INST rule shown in Figure 7 of [60], and say $\lambda X x Y.m[x[P_1|X]|Y] \xrightarrow{\vec{M}_\sigma^\alpha \downarrow}_{\mathcal{A}} m[n[P_1|P_2]|P_3]$. Therefore, thanks to C$\lambda$ rule in Figure 8 of [60], $P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha} m[n[P_1|P_2]|P_3]$, and by STRCNG rule $P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha} Q$.

- Assume that $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ by INPAR rule. This means that $\alpha = in\ m$, $P = P_1|Q_1$, $C_\epsilon^{in\ m}[-] = x[-|X_1]|m[X_2]$, $P_1 \xrightarrow{C_\epsilon^{in\ m}[-]}_{\mathcal{S}} P_\epsilon$ and $Q_\epsilon = P_\epsilon\{^{Q_1|X_1}/_{X_1}\}$. Moreover, the substitution σ has the following shape $\{^{P_2}/_{X_1}, ^n/_x, ^{P_3}/_{X_2}\}$, for some ambient name n and some processes P_1 and P_2.

  Let us consider the substitution $\sigma' = \{^{P_2|Q_1}/_{X_1}, ^n/_x, ^{P_3}/_{X_2}\}$. Note that $P_\epsilon\sigma' = Q_\epsilon\sigma \equiv Q$. Since $P_1 \xrightarrow{C_\epsilon^{in\ m}[-]}_{\mathcal{S}} P_\epsilon$, then $P_1 \xrightarrow{C_\epsilon^{in\ m}[-]\sigma'}_{\mathcal{S}_{\mathcal{I}}} P_\epsilon\sigma'$. Therefore, by applying the induction hypothesis, we have $P_1 \xrightarrow{in\ m \downarrow \vec{M}_{\sigma'}^\alpha}_{\mathcal{CA}} P_\epsilon\sigma'$, where $\vec{M}_{\sigma'}^\alpha = P_2|Q_1, n, P_3$.

  We have to show that $P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q$, where $\vec{M}_\sigma^\alpha = P_2, n, P_3$. We know that $P_1 \xrightarrow{in\ m \downarrow \vec{M}_{\sigma'}^\alpha}_{\mathcal{CA}} P_\epsilon\sigma'$. This means that $P_1 \xrightarrow{in\ m}_{\mathcal{C}} A$ and $A \xrightarrow{\vec{M}_{\sigma'}^\alpha \downarrow}_{\mathcal{A}} P_\epsilon\sigma'$. Since $P_1 \xrightarrow{in\ m}_{\mathcal{C}} A$, thanks to $||$IN rule of Figure 6 in [60], we have $P_1|Q_1 \xrightarrow{in\ m}_{\mathcal{C}} \lambda X.A(Q_1|X)$. It is easy to check that if $A \xrightarrow{\vec{M}_{\sigma'}^\alpha \downarrow}_{\mathcal{A}} P_\epsilon\sigma'$, then we also have $\lambda X.A(Q_1|X) \xrightarrow{\vec{M}_\sigma^\alpha \downarrow}_{\mathcal{A}} P_\epsilon\sigma'$. Therefore, by applying Cλ rule in Figure 8 of [60], we obtain $P_1|Q_1 \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} P_\epsilon\sigma'$, and by STRCNG rule $P_1|Q_1 \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q$.

- Assume that $P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$ by COIN rule. This means that $\alpha = \overline{[in\ m]}$, $P = m[P_1]$, $C_\epsilon^{\overline{[in\ m]}}[-] = -|x[in\ m.X_1|X_2]$ and $Q_\epsilon = m[x[X_1|X_2]|P_1]$. Moreover, the substitution σ has the following shape $\{^{P_2}/_{X_1}, ^{P_3}/_{X_2}, ^n/_x\}$, for some ambient name n and some processes P_1 and P_2. So, we have $Q \equiv Q_\epsilon\sigma = m[n[P_2|P_3]|P_1]$.

  We have to show that $P \xrightarrow{\overline{[in\ m]} \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q_\epsilon\sigma$, where $\vec{M}_\sigma^\alpha = P_2, P_3, n$. It is easy to check that $P \xrightarrow{\overline{[in\ m]}} \lambda Z.Z(P_1)$, by COIN rule in Figure 6 of [60]. Moreover, by INST rule shown in Figure 7 of [60], $(\lambda Z.Z(P_1))(\lambda XYZx.m[x[Y|Z]|X]) \xrightarrow{\vec{M}_\sigma^\alpha \downarrow}_{\mathcal{A}} m[n[P_2|P_3]|P_1]$. Therefore, thanks to COINλ rule of Figure 8 of [60], we can conclude $P \xrightarrow{\overline{[in\ m]} \downarrow \vec{M}_\sigma^\alpha} m[n[P_2|P_3]|P_1]$, and so $P \xrightarrow{\overline{[in\ m]} \downarrow \vec{M}_\sigma^\alpha} Q$.

□

Appendix C

Proofs of Chapter 5

C.1 $\sim^{(W)BSS} = \sim^{(W)BS}$

The proof of Proposition 5.1 is analogous to the proof of Proposition 5.2.

In order to prove Proposition 5.2 we give two additional definitions of weak barbed saturated bisimulation and we prove that they are all equivalent.

Definition C.1 (Weak Barbed Saturated Bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* weak barbed saturated bisimulation *iff whenever* $P \,\mathcal{R}\, Q$*, then*

1. *if* $P \downarrow_o$ *then* $Q \Downarrow_o$*,*
2. $\forall C[-]$*, if* $C[P] \to^* P'$ *then* $C[Q] \to^* Q'$ *and* $P' \,\mathcal{R}\, Q'$*.*

Proposition C.1. Definition 5.9 and Definition C.1 coincide.

Proof. First of all notice that the second conditions of both definition coincide.

Now, let $\mathcal{R}$ be a symmetric relation that satisfies the Definition 5.9. Then $\mathcal{R}$ also satisfies the first condition of Definition C.1. Indeed, suppose that $P \;\mathcal{R}\; Q$. If $P \;\downarrow_o$, then $P \;\Downarrow_o$ and, since $\mathcal{R}$ satisfies Definition 5.9, then $Q \Downarrow_o$.

Now, let $\mathcal{R}$ be a symmetric relation that satisfies the Definition C.1. Then $\mathcal{R}$ also satisfies the first condition of Definition 5.9. Indeed, suppose that $P \;\mathcal{R}\; Q$. If $C[P] \;\Downarrow_o$ then there exists P' such that $C[P] \to^* P'$ and $P' \downarrow_o$. Since $\mathcal{R}$ satisfies the second condition of Definition C.1, then there exists Q' such that $C[Q] \to^* Q'$ and $P' \;\mathcal{R}\; Q'$. Now, since $\mathcal{R}$ satisfies the first condition of Definition C.1, then $Q' \Downarrow_o$, i.e., $Q' \to^* Q'' \downarrow_o$. So, $C[Q] \to^* Q' \to^* Q'' \downarrow_o$, i.e., $C[Q] \Downarrow_o$. □

Definition C.2 (Weak Barbed Saturated Bisimulation)**.** *A symmetric relation* $\mathcal{R}$ *is a* weak barbed saturated bisimulation *iff whenever* $P \,\mathcal{R}\, Q$*, then*

1. *if* $P \downarrow_o$ *then* $Q \Downarrow_o$*,*
2. $\forall C[-]$*, if* $C[P] \to P'$ *then* $C[Q] \to^* Q'$ *and* $P' \,\mathcal{R}\, Q'$*.*

Proposition C.2. Definition C.1 and Definition C.2 coincide.

Proof. First of all notice that the first conditions of both definitions coincide.

Now, let $\mathcal{R}$ be a symmetric relation that satisfies the Definition C.1. Then $\mathcal{R}$ also satisfies the second condition of Definition C.2. Indeed, suppose that $P \,\mathcal{R}\, Q$. If $C[P] \to P'$, then also $C[P] \to^* P'$ and, since $\mathcal{R}$ satisfies Definition C.1, then there exists Q' such that $C[Q] \to^* Q'$ and $P' \,\mathcal{R}\, Q'$.

Now, let $\mathcal{R}$ be a symmetric relation that satisfies the Definition C.2. Then $\mathcal{R}$ also satisfies the second condition of Definition C.1. Indeed, suppose that $P \,\mathcal{R}\; Q$. If $C[P] \to^* P'$ then there exist $P'_1 \dots P'_n$ such that $P'_n = P'$ and $C[P] \to P'_1 \to \dots \to P'_n$. Since $\mathcal{R}$ satisfies the second condition of Definition C.2, then there exist $Q'_1 \dots Q'_n$ such that $C[Q] \to^* Q'_1 \to^* \dots \to^* Q'_n$ and $\forall i \in 1 \dots n$, $P'_i \;\mathcal{R}\; Q'_i$. Thus, $C[Q] \to^* Q'_n$ such that $P' = P'_n \,\mathcal{R}\, Q'_n$. □

Now we can prove Proposition 5.2.

Proof of Proposition 5.2. We prove that $\sim^{WBSS} \subseteq \sim^{WBS}$, showing that the contextual closure S of weak barbed semi-saturated bisimilarity

$$S = \{\langle C[P], C[Q]\rangle \mid P \sim^{WBSS} Q,\ C \in \mathbf{C}\}$$

is a weak barbed saturated bisimulation with respect to Definition C.2. Suppose that $C[P]\, S\, C[Q]$. The first condition of Definition C.2 is trivially satisfied, since $P \sim^{WBSS} Q$.

$$\begin{tikzcd}
 & I_6 & & \\
I_4 \arrow[ur, "f"] & & I_5 \arrow[ul, "d''"'] & \\
 & I_2 \arrow[ul, "C"] \arrow[ur, "g"] & & I_3 \arrow[ul, "d'"'] \arrow[uull, bend right, "d"'] \\
 & & 0 \arrow[ul, "P"] \arrow[ur, "l"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
 & I_6 & & \\
I_4 \arrow[ur, "f"] & & I_5 \arrow[ul, "d''"'] & \\
 & I_2 \arrow[ul, "C"] \arrow[ur, "g"] & & I_3 \arrow[ul, "e"'] \\
 & & 0 \arrow[ul, "Q"] \arrow[ur, "l'"'] &
\end{tikzcd}$$

(ii)

Suppose that $f[C[P]] \to P'$.

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 lower square of diagram (i) and then $P \to^{g}_{IPO} d'[r]$. Since $P \sim^{WBSS} Q$ we have that $g[Q] \to^{*} Q'$ with $d'[r] \sim^{WBSS} Q'$. Now, since $d''[-]$ is reactive, we have that $f[C[Q]] = d''[g[Q]] \to^{*} d''[Q']$. Since $d'[r] \sim^{WBSS} Q'$, then $P' = d''[d'[r]]\, S\, d''[Q']$.

In order to prove that $\sim^{WBS} \subseteq \sim^{WBSS}$ it is enough to consider Definition 5.9 and to observe that if $C[P] \downarrow_o$ then $C[P] \Downarrow_o$ and that if $P \to^{C[-]}_{IPO} P'$ then $C[P] \to^{*} P'$. □

C.2 Mobile Ambients Barbs are Contextual

Before proving Proposition 5.4, we recall the mobile ambients barbs and mobile ambients contexts.

Given a mobile ambients process P, we have $P \downarrow_n$ if $P \equiv (\nu A)(n[Q]|R)$ and $n \notin A$, for some processes Q and R and a set of restricted names A.

Mobile ambients contexts are terms of the extended syntax with a hole $-$, formally, they are generated by the following grammar:

$$C[-] ::= -, C[-]|R, (\nu n)C[-], n[C[-]]$$

where R is an arbitrary process.

Now we show the proof of Proposition 5.4.

Proof of Proposition 5.4. We only prove that mobile ambients barbs are strong contextual barbs. The proof for the weak case is similar.

We have to show that whenever $P \downarrow_n$ implies $Q \downarrow_n$ then for all context $C[-]$, $C[P] \downarrow_n$ implies $C[Q] \downarrow_n$.

We assume $P \downarrow_n$ implies $Q \downarrow_n$ and we prove that $\forall C[-]$, $C[P] \downarrow_n$ implies $C[Q] \downarrow_n$. The proof proceeds by structural induction on the context $C[-]$.

- Assume that $C[-] = -$. It means that $C[P] = P$ and $C[Q] = Q$. Since $P \downarrow_n$ implies $Q \downarrow_n$, it is obvious that $C[P] \downarrow_n$ implies $C[Q] \downarrow_n$.

- Assume that $C[-] = C'[-]|R$. It means that $C[P] = C'[P]|R$ and $C[Q] = C'[Q]|R$. If $C'[P]|R \downarrow_n$, then either $C'[P] \downarrow_n$ or $R \downarrow_n$. If $C'[P] \downarrow_n$, then we can apply the induction hypothesis and say that $C'[Q] \downarrow_n$, and hence also $C'[Q]|R \downarrow_n$. In the case of $R \downarrow_n$, it obvious that also $C'[Q]|R \downarrow_n$.

- Assume that $C[-] = (\nu m)C'[-]$. It means that $C[P] = (\nu m)C'[P]$ and $C[Q] = (\nu m)C'[Q]$. If $(\nu m)C'[P] \downarrow_n$, then $C'[P] \downarrow_n$ and $n \neq m$. Therefore, we can apply the induction hypothesis and say that $C'[Q] \downarrow_n$, and hence also $(\nu m)C'[Q] \downarrow_n$.
- Assume that $C[-] = m[C'[-]]$. It means that $C[P] = m[C'[P]]$ and $C[Q] = m[C'[Q]]$. If $m[C'[P]] \downarrow_n$, then $n = m$. Therefore, it is obvious that $C[Q] \downarrow_n$.

$\square$

Bibliography

[1] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *TCS*, 195(2):291–324, 1998.

[2] P. Baldan, A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Concurrent semantics of algebraic graph transformation. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, pages 107–187. World Scientific, 1999.

[3] P. Baldan, H. Ehrig, and B. König. Composition and decomposition of DPO transformations with borrowed context. In *ICGT'06*, volume 4178 of *LNCS*, pages 153–167. Springer, 2006.

[4] H. Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[5] J. Bénabou. Introduction to bicategories. In *Midwest Category Seminar*, volume 47 of *Lectures Notes in Mathematics*, pages 1–77. Springer, 1967.

[6] F. Bonchi. *Abstract Semantics by Observable Contexts*. PhD thesis, Department of Informatics, University of Pisa, 2008.

[7] F. Bonchi, F. Gadducci, and B. König. Process bisimulation via a graphical encoding. In *ICGT'06*, volume 4178 of *LNCS*, pages 168–183. Springer, 2006.

[8] F. Bonchi, F. Gadducci, and G. V. Monreale. RPO semantics for the calculus of mobile ambients. *Mathematical Structures in Computer Science*, submitted.

[9] F. Bonchi, F. Gadducci, and G. V. Monreale. Labelled transitions for mobile ambients (as synthesized via a graphical encoding). In *EXPRESS'08*, volume 242(1) of *ENTCS*, pages 73–98, 2009.

[10] F. Bonchi, F. Gadducci, and G. V. Monreale. On barbs and labels in reactive systems. In *SOS'09*, volume 18 of *EPTCS*, pages 46–61, 2009.

[11] F. Bonchi, F. Gadducci, and G. V. Monreale. Reactive systems, barbed semantics, and the mobile ambients. In *FOSSACS'09*, volume 5504 of *LNCS*, pages 272–287. Springer, 2009.

[12] F. Bonchi, F. Gadducci, G. V. Monreale, and U Montanari. Saturated LTSs for adhesive rewriting systems. In *ICGT'10*, volume 6372 of *LNCS*. Springer, 2010.

[13] F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *Logic in Computer Science*, pages 69–80. IEEE Computer Society, 2006.

[14] M. Boreale, R. De Nicola, and R. Pugliese. Asynchronous observations of processes. In M. Nivat, editor, *FoSSaCS'98*, volume 1378 of *LNCS*, pages 95–109. Springer, 1998.

[15] R. Bruni, F. Gadducci, and U. Montanari. Normal forms for algebras of connections. *TCS*, 286(2):247–292, 2002.

[16] L. Cardelli and A. Gordon. Mobile ambients. *TCS*, 240(1):177–213, 2000.

[17] I. Castellani and M. Hennessy. Testing theories for asynchronous languages. In V. Arvind and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1530 of *LNCS*, pages 90–101. Springer, 1998.

[18] P. Cenciarelli, I. Talamo, and A. Tiberi. Ambient graph rewriting. In N. Martì-Oliet, editor, *Rewriting Logic and its Applications*, volume 117 of *ENTCS*, pages 335–351. Elsevier, 2005.

[19] A. Corradini and F. Gadducci. An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures*, 7:299–331, 1999.

[20] A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26(3/4):241–265, 1996.

[21] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 163–245. World Scientific, 1997.

[22] Grohmann. D. and M. Miculan. Reactive systems over directed bigraphs. In *CONCUR '07*, volume 4703 of *LNCS*, pages 380–394. Springer, 2007.

[23] P. Di Gianantonio, F. Honsel, and M. Lenisa. Rpo, second-order contexts, and λ-calculus. In *FoSSaCS '08*, volume 4962 of *LNCS*, pages 334–349. Springer, 2008.

[24] F. Drewes, A. Habel, and H.-J. Kreowski. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 95–162. World Scientific, 1997.

[25] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, 2006.

[26] H. Ehrig, G. Engels, Kreowski H. J., U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1-3. World Scientific, 1997-1999.

[27] H. Ehrig and B. König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In *FoSSaCS'04*, volume 2987 of *LNCS*, pages 151–166. Springer, 2004.

[28] H. Ehrig and B. König. Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. *Mathematical Structures in Computer Science*, 16(6):1133–1163, 2006.

[29] H. Ehrig, M. Pfender, and Schneider H. J. Graph-grammars: an algebraic approach. In *Switching and Automata Theory*, pages 167–180. IEEE Computer Society Press, 1973.

[30] G. Ferrari, U. Montanari, and E. Tuosto. A LTS semantics of ambients via graph synchronization with mobility. In A. Restivo, S. Ronchi Della Rocca, and L. Roversi, editors, *Italian Conference on Theoretical Computer Science*, volume 2202 of *LNCS*, pages 1–16. Springer, 2001.

[31] C. Fournet and Gonthier. G. A hierarchy of equivalences for asynchronous calculi. In *ICALP*, pages 844–855, 1998.

[32] F. Gadducci. Graph rewriting for the π-calculus. *Mathematical Structures in Computer Science*, 17(3):407–437, 2007.

[33] F. Gadducci and R. Heckel. An inductive view of graph transformation. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques*, volume 1376 of *LNCS*, pages 219–233. Springer, 1997.

[34] F. Gadducci and G. V. Monreale. A decentralized implementation of mobile ambients. In *ICGT'08*, volume 5214 of *LNCS*, pages 115–130. Springer, 2008.

[35] F. Gadducci and G. V. Monreale. A decentralised graphical implementation of mobile ambients. *Journal of Logic and Algebraic Programming*, 80(2):113–136, 2011.

[36] F. Gadducci and U. Montanari. A concurrent graph semantics for mobile ambients. In S. Brookes and M. Mislove, editors, *Mathematical Foundations of Programming Semantics*, volume 45 of *ENTCS*. Elsevier Science, 2001.

[37] A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. *Mathematical Structures in Computer Science*, 13(3):371–408, 2003.

[38] D. Grohmann and M. Miculan. Deriving barbed bisimulations for bigraphical reactive systems. In *ICGT'08 - Doctoral Symposium*, volume 16 of *ECEASST*. EASST, 2008.

[39] A. Habel, J. Müller, and P. Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.

[40] K. Honda and N. Yoshida. On reduction-based process semantics. *TCS*, 151(2):437–486, 1995.

[41] O. Jensen and R. Milner. Bigraphs and mobile processes. Technical Report 580, Computer Laboratory, University of Cambridge, 2003.

[42] O. H. Jensen and R. Milner. Bigraphs and transitions. In *POPL*, pages 38–49, 2003.

[43] O.H. Jensen. *Mobile Processes in Bigraphs*. PhD thesis, University of Aalborg, 2008.

[44] S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(3):511–545, 2005.

[45] J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Concurrency Theory*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[46] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *POPL*, pages 352–364, 2000.

[47] M. Merro and F. Zappa Nardelli. Behavioral theory for mobile ambients. *Journal of the ACM*, 52(6):961–1023, 2005.

[48] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[49] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[50] R. Milner. Bigraphs for petri nets. In *Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 686–701. Springer, 2004.

[51] R. Milner. Pure bigraphs: Structure and dynamics. *Information and Computation*, 204(1):60–122, 2006.

[52] R. Milner. Local bigraphs and confluence: Two conjectures. *ENTCS*, 175(3):65–73, 2007.

[53] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, 1992.

[54] R. Milner and D. Sangiorgi. Barbed bisimulation. In *ICALP*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[55] N. Mylonakis and F. Orejas. Another fully abstract graph semantics for the ambient calculus. Presented at *Graph Transformation - Verification and Concurrency 2007*.

[56] D. M. R. Park. Concurrency and automata on infinite sequences. volume 104 of *LNCS*. Springer, 1980.

[57] G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

[58] J. Rathke, V. Sassone, and P. Sobociński. Semantic barbs and biorthogonality. In *FoSSaCS'07*, volume 4423 of *LNCS*, pages 302–316. Springer, 2007.

[59] J. Rathke and P. Sobociński. Deconstructing behavioural theories of mobility. In *Theoretical Computer Science*, volume 273 of *LNCS*, pages 507–520. Springer, 2008.

[60] J. Rathke and P. Sobociński. Deriving structural labelled transitions for mobile ambients. In *Concurrency Theory*, volume 5201 of *LNCS*, pages 462–476. Springer, 2008.

[61] V. Sassone and P. Sobociński. Deriving bisimulation congruences using 2-categories. *Nordic Journal of Computing*, 10(2):163–183, 2003.

[62] V. Sassone and P. Sobociński. A congruence for Petri nets. In *Petri Nets and Graph Transformation*, volume 127 of *ENTCS*, pages 107–120. Elsevier, 2005.

[63] V. Sassone and P. Sobociński. Reactive systems over cospans. In *Logic in Computer Science*, pages 311–320, 2005.

[64] P. Sobociński. *Deriving bisimulation congruences from reduction systems*. PhD thesis, BRICS, Department of Computer Science, University of Aaurhus, 2004.