Math. Struct. in Comp. Science (2014), vol. 24, iss. 4, e240404, 59 pages. © Cambridge University Press 2014
doi:10.1017/S0960129512000333

RPO semantics for mobile ambients[†]

FILIPPO BONCHI[‡], FABIO GADDUCCI[§] and
GIACOMA VALENTINA MONREALE[§]

[‡]*ENS Lyon, Université de Lyon,*
LIP (UMR 5668 CNRS ENS Lyon UCBL INRIA),
Lyon, France
Email: fibonchi@di.unipi.it
[§]*Dipartimento di Informatica, Università di Pisa,*
Pisa, Italy
Email: {vale;fabio}@di.unipi.it

Received 24 February 2011; revised 6 January 2012

In this paper we focus on the synthesis of labelled transition systems (LTSs) for process calculi using Mobile Ambients (MAs) as a testbed. Our proposal is based on a graphical encoding: a process is mapped into a graph equipped with interfaces such that the denotation is fully abstract with respect to the standard structural congruence. Graphs with interfaces are amenable to the synthesis mechanism based on borrowed contexts (BCs), which is an instance of relative pushouts (RPOs). The BC mechanism allows the effective construction of an LTS that has graphs with interfaces as states and labels, and such that the associated bisimilarity is a congruence. We focus here on the analysis of an LTS over processes as graphs with interfaces: we use the LTS on graphs to recover an LTS directly defined over the structure of MA processes and define a set of SOS inference rules capturing the same operational semantics.

1. Introduction

Process calculi are a powerful formalism for the specification of concurrent and distributed systems. Their flexibility is proved by the wide range of different domains in which they have been applied. Their expressiveness is guaranteed by the comparison with standard (for example, functional) computational frameworks. Finally, their usability is supported by well-defined methodologies for the presentation of their semantics.

The operational semantics is usually defined by means of labelled transitions systems (LTSs): a set of states of the system, plus a labelled relation describing its transitions. This makes it easy to provide a behavioural semantics based on observations by looking at the labels of the evolutions that might be performed by the system.

These LTSs are usually obtained inductively by means of a set of inference rules. Nevertheless, obtaining such a compact presentation of an LTS is often complex and relies on the ingenuity of the researcher, which is nowadays tested by the increasing

[†] This research was partially supported by the EU FP7-ICT IP ASCENs (IP 257414) and by the MIUR PRIN SisteR (PRIN 20088HXMYN).

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

complexity of the calculi. Hence, it has become customary, particularly following Milner's treatment of π-calculus (Milner 1999), to present the semantics of a calculus by a reduction semantics: an unlabelled relation (also generated by a set of rules) modulo a structural congruence equating those processes that intuitively represent the same distributed system.

These considerations can be applied *verbatim* to one of the most recent and fruitful proposals, *viz*. the calculus of mobile ambients (MAs) – see Cardelli and Gordon (2000). The analogy between ambients and network domains, which has been studied since the introduction of the calculus, and between ambients and molecular environments, which is often exploited in system biology, has made MAs a centrepiece of recent applications and developments of the process calculi paradigm.

It is thus baffling that the calculus has been so resilient to the introduction of an observational semantics. This is probably because the set of rules defining the original reduction semantics of MAs is already rather complex. For example, the system evolution for the 'exporting' of a process P out of an ambient named n is represented by the rule

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

This rule needs to carry around with it the occurrences of processes Q and R, which denote the context into which the instance of the rule has to be mapped. The need for such rich contextual information makes it difficult to obtain a satisfying observational semantics. Following the early proposals in Gordon and Cardelli (2003) and Ferrari *et al.* (2001), the only work we are aware of is Merro and Zappa Nardelli (2005) and, more recently, the work originally presented in Rathke and Sobociński (2008b) and then partly modified in Rathke and Sobocinski (2010).

Apart from Rathke and Sobociński (2008b; 2010), the other proposals are not based on a methodological assumption. Indeed, it was only recently that a series of papers addressed the problem of automatically synthesising an LTS from the reduction semantics of a calculus, further guaranteeing that the derived observational semantics is a congruence. The most successful technique so far was proposed in Leifer and Milner (2000) and is based on the notion of a *relative pushout* (RPO): it captures in an abstract setting the intuitive notion of a minimal context into which a process has to be inserted in order to enable a reduction to occur.

However, proving that a calculus satisfies the requirements needed for applying the RPOs technique is often a daunting task. One way out of this impasse is represented by the graphical encodings of processes, which turns the structural congruence into a graph isomorphism. Graph formalisms are usually amenable to the RPO technique, and once the processes of a calculus have been encoded as graphs, an LTS can be distilled. Indeed, it does not seem to be mere chance that the main source of examples concerning RPOs have been *bigraphs* (Milner 2006).

It is noteworthy that when the reduction relation over graphs is defined using the double pushout (DPO) approach (Baldan *et al.* 1999), these graphs are amenable to the *borrowed context* (BC) technique (Ehrig and König 2006), which was developed by Ehrig and König, and offers an algorithmic solution for calculating the minimal contexts enabling a graph transformation rule. Indeed, graphs form an *adhesive* category (Lack and Sobociński 2005),

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

and thus, through the so-called *cospan* construction (Sassone and Sobociǹski 2005), BCs and RPOs can be proved to be coincident notions (Sobociński 2004).

Hence, the approach pursued in this and papers such as Bonchi *et al.* (2006a) and Gadducci and Montanari (2005) is quite straightforward: for a given calculus, we find a graphical encoding such that the structural congruence is preserved, and then the reduction semantics is captured by a set of graph transformation rules specified using the DPO approach. Hence, an LTS for the calculus can be distilled. This is the approach that allowed the only successful application so far of the RPO technique to the set of recursive processes of a calculus, while still recovering the standard bisimulation congruence, even if it was for one of the simplest calculi, *viz*. Milner's CCS (Milner 1989).

In the current paper, we exploit the graphical encoding for MAs proposed in Gadducci and Monreale (2008) to distill an LTS on (processes encoded as) graphs. This LTS is used to infer a set of SOS rules defined on the processes of MAs, which is useful for comparing it with the alternative solution proposed in Rathke and Sobociński (2008b), which was also inspired by the RPO technique, thereby discovering many similarities and a few substantial differences – see the concluding section of the paper (Section 11) for a summary.

1.1. *Organisation of the paper*

Section 2 provides a summary of a few notions concerning MAs, and Section 3 introduces typed (hyper-)graphs with interfaces. In Section 4, we recall the DPO approach to graph rewriting and the associated BC technique for distilling an LTS, and in Section 5, we discuss a graphical encoding for MAs processes.

In Section 6, we describe the simulation of process reduction by DPO rewriting, which is needed in Section 7 for the presentation of an LTS for graphs with interfaces representing MA processes, which is then exploited in Section 8 to introduce an LTS defined directly over MAs processes.

We then present a novel description of the distilled LTS in Section 9 by means of a set of SOS-style inference rules, which is finally used in Section 10 to prove the correspondence between our proposal and Rathke and Sobociński's.

We give a summary of our conclusions in Section 11, and mention related work together with some ideas for future work in the area.

Finally, a series of appendices contain proofs of some of the theorems from earlier sections in the paper and a standard SOS presentation of the LTS.

1.2. *Previous work*

The work in the current paper builds on Bonchi *et al.* (2009a): the presentation of the individual sections has been tightened and reshaped; detailed proofs have been included (see Appendix A and Appendix B); and a few new examples have been added. Two new sections, Sections 9 and 10, are partly drawn (with considerable restructuring) from Bonchi *et al.* (2009c). The SOS-like presentation of the distilled LTS in Section 9 is a fundamental feature of the paper and is used to reason about the structural properties

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

$$P ::= 0, n[P], M.P, (\nu n)P, P_1|P_2, X, x[P]$$

$$M ::= in\ n, out\ n, open\ n$$

Table 1. *Extended syntax of mobile ambients.*

of the operational semantics: its presence strengthens our claims about the practical relevance of our proposal, which is further confirmed by the purely SOS presentation of the LTS given in Appendix C. Our new presentation allows a simple proof of the coincidence between our LTS and Rathke and Sobociński's, as shown in Section 10, and this coincidence provides further evidence of the soundness of the operational semantics we propose here for MAs.

2. Mobile ambients

In this section we briefly recall the finite, communication-free fragment of mobile ambients (Cardelli and Gordon 2000), together with its structural equivalence and reduction semantics.

Note that to help the presentation of the operational semantics in Section 8, we introduce an extended syntax that allows us to build processes containing *process* and *name variables*.

Definition 2.1 (extended processes). Let:

— $\mathcal{N}$ be a set of *names* ranged over by $m, n, u, \ldots$;
— $\mathcal{X} = \{X, Y, \ldots\}$ be a set of *process variables*; and
— $\mathcal{V} = \{x, y, \ldots\}$ be a set of *name variables*.

Then an *extended process* is a term generated by the syntax in Table 1.

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 2.2 (pure and linear processes). An extended process is a *pure* process if no process or name variable occurs in it, and it is a *linear* process if no process or name variable occurs in it more than once.

We let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of pure processes and $P_\epsilon, Q_\epsilon, R_\epsilon, \ldots$ range over the set $\mathcal{P}_\epsilon$ of linear processes. We use the standard definitions for both the set of free names of a pure process P, denoted by $fn(P)$, and for α-convertibility with respect to the restriction operators. Variables carry no names, hence

$$fn(x[P_\epsilon]) = fn(P_\epsilon)$$
$$fn(X) = \varnothing.$$

The sets of name and process variables are defined as expected and denoted by $nv(P_\epsilon)$ and $pv(P_\epsilon)$, respectively.

Moreover, we consider a family of *substitutions*, which may replace a process or name variable with a pure process or name, respectively. Substitutions avoid name capture: for

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

$$
\begin{array}{rll}
P|Q \equiv Q|P & & \\
(P|Q)|R \equiv P|(Q|R) & & \\
(vn)(vm)P \equiv (vm)(vn)P & & \\
(vn)(P|Q) \equiv P|(vn)Q & & \text{if } n \notin fn(P) \\
P \equiv Q \Rightarrow n[P] \equiv n[Q] & & \\
(vn)m[P] \equiv m[(vn)P] & & \text{if } n \neq m \\
P \equiv Q \Rightarrow M.P \equiv M.Q & & \\
P|0 \equiv P & & \\
P \equiv Q \Rightarrow (vn)P \equiv (vn)Q & & \\
(vn)M.P \equiv M.(vn)P & & \text{if } n \notin fn(M) \\
P \equiv Q \Rightarrow P|R \equiv Q|R & & \\
(vn)P \equiv (vm)(P\,\{^{m}/_{n}\}) & & \text{if } m \notin fn(P)
\end{array}
$$

Table 2. *Structural congruence on pure processes.*

$$
\begin{array}{rl}
\text{(InRed)} & n[in\ m.P|Q]|m[R] \rightarrow m[n[P|Q]|R] \\
\text{(ResRed)} & P \rightarrow Q \Rightarrow (vn)P \rightarrow (vn)Q \\
\text{(OutRed)} & m[n[out\ m.P|Q]|R] \rightarrow n[P|Q]|m[R] \\
\text{(AmbRed)} & P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q] \\
\text{(OpenRed)} & open\ n.P|n[Q] \rightarrow P|Q \\
\text{(ParRed)} & P \rightarrow Q \Rightarrow P|R \rightarrow Q|R
\end{array}
$$

Table 3. *Reduction relation on pure processes.*

a pure process P, the expression

$$(vn)(vm)(m[X]|x[0])\left\{^{m}/_{x}, ^{n[P]}/_{X}\right\}$$

corresponds to the pure process

$$(vp)(vq)(q[n[P]]|m[0])$$

for names

$$p, q \notin \{m\} \cup \mathit{fn}(n[P]).$$

The semantics exploits a *structural congruence*, denoted by $\equiv$, which is the least equivalence on pure processes that satisfies the equations in Table 2. The congruence relates processes that, intuitively, specify the same system, up-to a syntactical rearrangement of its components, and it is then used to define a *reduction relation*, which we introduce below and denote by $\rightarrow$. The relation describes the evolution of processes over time: $P \rightarrow Q$ means that P reduces to Q, that is, P executes a computational step and evolves into Q.

Definition 2.3 (reduction relation). The *reduction relation* $\mathcal{R}_{amb} \subseteq \mathcal{P} \times \mathcal{P}$ is the smallest relation that is closed under $\equiv$ and generated by the rules in Table 3.

Our chosen congruence differs slightly from the standard one by dropping

$$(vn)0 \equiv 0$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

and adding

$$(\nu n)M.P \equiv M.(\nu n)P,$$

thereby allowing a restriction to enter a capability. The reduction semantics is not substantially changed by this: the equality induced by the new axiom holds in all the behavioural equivalences for mobile ambients that we are aware of. In particular, two processes that are structurally congruent are *reduction barbed* congruent (Merro and Zappa Nardelli 2005).

Example 2.1. As a running example, we will consider processes

$$Q = n[in\ m.0]|m[out\ m.0]$$
$$P = (\nu n)Q.$$

The application of the (InRed) axiom to Q results in

$$m[n[0]|out\ m.0],$$

so we may apply the rule (ResRed) to P and conclude

$$P \to (\nu n)(m[n[0]|out\ m.0]).$$

3. Graphs and their extension with interfaces

In this section we will recall a few definitions concerning typed (hyper-)graphs, and their extension with *interfaces* – see, for example, Corradini and Gadducci (1999) for a more detailed introduction.

Definition 3.1 (typed graphs). A (*hyper-*)*graph* is a four-tuple $\langle N, E, s, t\rangle$ for N and E the sets of nodes and (hyper-)edges and $s, t : E \to N^*$ the source and target functions. A *graph morphism* is a pair of functions $\langle f_N, f_E\rangle$ preserving the source and target.

Let T be a graph. A *typed graph* G over T is a pair $\langle |G|, t_G\rangle$ for $|G|$ a graph and $t_G : |G| \to T$ the typing morphism. A *T-typed graph morphism* is a graph morphism $f : |G_1| \to |G_2|$ between the underlying graphs preserving the typing.

The category of graphs typed over T is denoted by T-**Graph**.

Definition 3.2 (graphs with interfaces). Let J and K be typed graphs. A *graph with input interface J and output interface K* is a triple $\mathbb{G} = \langle j, G, k\rangle$ for G a typed graph and $j : J \to G$ and $k : K \to G$ the *input* and *output* morphisms, respectively. Let $\mathbb{G}$ and $\mathbb{H}$ be graphs with the same interfaces. An *interface graph morphism* is a typed graph morphism $f : G \to H$ between the underlying typed graphs preserving the input and output.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

We let $J \xrightarrow{j} G \xleftarrow{k} K$ denote a graph with interfaces J and K. If the interfaces J and K are *discrete*, that is, they only contain nodes, we represent them by sets – if K is the empty set, we often denote a graph with interfaces as a graph morphism $J \to G$[†].

In order to define the encoding, we will need some operators on graphs with discrete interfaces. We rely here on the proposal in Gadducci and Monreale (2008), which should be consulted for further details.

Definition 3.3 (two operators). Let

$$\mathbb{G} = I \xrightarrow{j} G \xleftarrow{k} K$$
$$\mathbb{G}' = K \xrightarrow{j'} G' \xleftarrow{k'} J$$

be graphs with discrete interfaces. Their *sequential composition* is the graph with discrete interfaces

$$\mathbb{G} \circ \mathbb{G}' = I \xrightarrow{j''} G'' \xleftarrow{k''} J,$$

for G'' the disjoint union $G \uplus G'$, modulo the equivalence induced by $k(x) = j'(x)$ for all $x \in N_K$, with the uniquely induced arrows j'' and k''.

Let

$$\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$$
$$\mathbb{G}' = J' \xrightarrow{j'} G' \xleftarrow{k'} K'$$

be graphs with discrete, compatible interfaces[‡]. Their *parallel composition* is the graph with discrete interfaces

$$\mathbb{G} \otimes \mathbb{G}' = (J \cup J') \xrightarrow{j''} G'' \xleftarrow{k''} (K \cup K'),$$

for G'' the disjoint union $G \uplus G'$, modulo the equivalence induced by $j(x) = j'(x)$ for all $x \in N_J \cap N_{J'}$, and $k(y) = k'(y)$ for all $y \in N_K \cap N_{K'}$, and j'', k'' the uniquely induced arrows.

The sequential composition $\mathbb{G} \circ \mathbb{G}'$ takes the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{G}'$ and glues the outputs of $\mathbb{G}$ with the corresponding inputs of $\mathbb{G}'$. The parallel composition $\mathbb{G} \otimes \mathbb{G}'$ takes the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{G}'$, and glues the inputs (outputs) of $\mathbb{G}$ with the corresponding inputs (outputs) of $\mathbb{G}'$. Both operations are defined on 'concrete' graphs, even if the result is independent of the chosen representatives.

[†] We often refer implicitly to a graph with interfaces as the representative of its isomorphism class and use the same symbols to denote it and its components. Moreover, in the discrete interfaces case, we refer to the nodes in the image of the input and output morphisms as inputs and outputs, respectively.

[‡] That is, any node in $N_J \cap N_{J'}$ has the same type in J and J' (and similarly for $N_K \cap N_{K'}$).

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

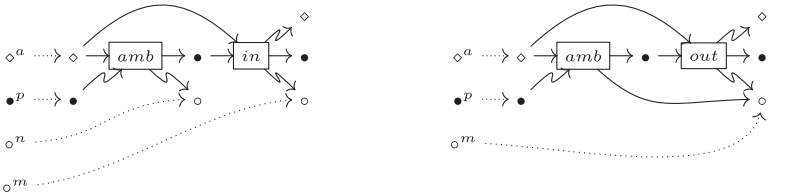


Fig. 1. Graphs with interfaces $\mathbb{G}$ and $\mathbb{G}'$ (from left to right).

Example 3.1. Consider the two graphs with interfaces

$$\mathbb{G} = \{a, p, n, m\} \to G \leftarrow \varnothing$$
$$\mathbb{G}' = \{a, p, m\} \to G' \leftarrow \varnothing$$

in Figure 1. As we will explain in Section 5, these two graphs represent the graphical encodings of the MA processes

$$Q_1 = n[in\ m.0]$$
$$Q_2 = m[out\ m.0]$$

respectively. For the moment, we will ignore how these encodings are obtained and just 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*, and, 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 input interfaces of the two graphs, and processes (sub-processes) are represented by graphs (sub-graphs) that have as root a pair of nodes $\langle \bullet, \diamond \rangle$. Only the root nodes $\langle \bullet, \diamond \rangle$ of the graphs representing the processes Q_1 and Q_2 are in the input interfaces of the corresponding graphs. Furthermore, each sub-process is represented by a sub-graph that has a different $\bullet$ root node, though sub-graphs representing different sub-terms sometimes share the $\diamond$ root node – we shall see later why this occurs.

The two graphs $\mathbb{G}$ and $\mathbb{G}'$ have compatible interfaces. Indeed, it is easy to check that the type of the nodes belonging to both input interfaces coincides. Therefore, it is possible to compute the parallel composition of the graphs $\mathbb{G}$ and $\mathbb{G}'$ to give the graph with interfaces shown in Figure 2. It is easy to see that it is obtained by making the respective unions of the input and output interfaces and the disjoint union of G and G', and then gluing the root nodes of both graphs and the nodes representing the name m. As we will explain in Section 5, 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 Q_1 and Q_2, that is, the process $Q_1|Q_2$.

4. Graphs with interfaces and borrowed contexts

In this section we introduce the *double-pushout* (DPO) approach to the rewriting of graphs with interfaces, and its extension with *borrowed contexts* (BCs), as introduced in Ehrig and König (2006).

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

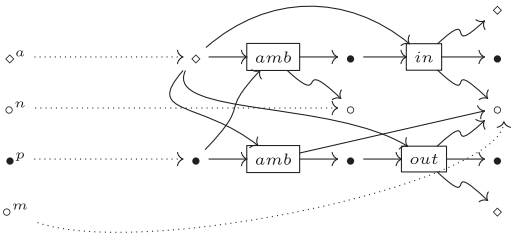


Fig. 2. Graph with interfaces $\mathbb{G} \otimes \mathbb{G}'$.

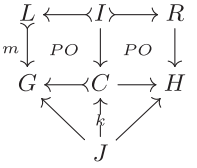


Fig. 3. A direct derivation.

Definition 4.1 (graph production). A T-typed *graph production* is a span of morphisms

$$L \overset{l}{\hookleftarrow} I \overset{r}{\longrightarrow} R$$

in T-**Graph** such that l is mono. A T-typed *graph transformation system* (GTS) $\mathcal{G}$ is a pair $\langle P, \pi \rangle$ where P is a set of production names and π is a function that assigns a T-typed production to each production name.

A production identifies those graph items that should be rewritten, and specifies how new items have to be inserted. The intuitive meaning is captured by the following definition.

Definition 4.2 (direct derivation). Let $J \to G$ and $J \to H$ be two graphs with interfaces and let

$$p : L \hookleftarrow I \to R$$

be 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 in Figure 3 such that the two squares are pushouts (PO), which is denoted by

$$J \to G \Longrightarrow J \to H.$$

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 impossible.

In these derivations, the left-hand side L of a production must then occur completely in G. In a *borrowed context* (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. These BCs are the 'smallest' contexts needed to obtain the image of L in G, and they may be used as suitable labels.

Definition 4.3 (rewriting with borrowed contexts). Given a production

$$p : L \hookleftarrow I \to R,$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

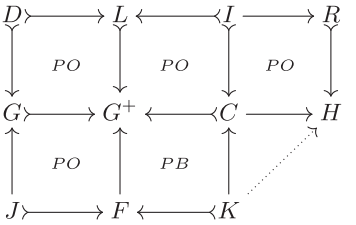


Fig. 4. A BC derivation.

a graph with interfaces $J \to G$ and a span of monos

$$d : G \leftarrowtail D \rightarrowtail L,$$

we say that $J \to G$ *reduces to* $K \to H$ with label

$$J \rightarrowtail F \leftarrowtail K$$

via p and d if there are graphs G^+ and C, and additional morphisms such that the diagram in Figure 4 commutes and the squares are either pushouts (PO) or pullbacks (PB). We write

$$J \to G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H,$$

and call it a *rewriting step with borrowed context*.

The upper left-hand square of the diagram in Figure 3 merges the left-hand side L and the graph 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 to produce the two other squares in the upper row. The pushout in the lower row gives the BC F, which is missing for 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.

Note that the two pushout complements needed in Definition 4.3, namely C and F, may not exist, in which case, the rewriting step is impossible.

5. Graphical encoding for processes

In this section we recall a graphical encoding for MA processes that was presented (with minor differences) in Gadducci and Monreale (2008). After describing a type graph (T_M, in Figure 5), we will define the encoding inductively by means of the composition operators introduced in Definition 3.3. This roughly corresponds to the standard construction of the tree for a term of an algebra: names are interpreted as variables, so they are mapped to graph leaves and can be shared.

Intuitively, a node of type $\circ$ represents an ambient name, while a graph that has as its 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 will need two different types of node to model processes by graphs because each graph

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

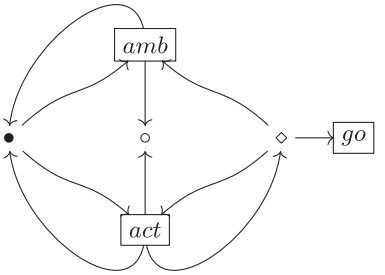


Fig. 5. The type graph T_M (for $act \in \{in, out, open\}$).

has to model both syntactical and activation dependencies between the operators of a process.

Each edge of T_M, except the go edge, simulates an MA operator. Note that the *act* edge stands for three edges, namely *in*, *out* and *open*. These edges simulate the capabilities of the calculus, and the *amb* edge simulates the ambient operator, while no edge simulates either restriction or parallel composition. The go edge is a syntactical device for detecting the 'entry' point for the computation, and is needed to simulate the MA reduction semantics: this allows us to avoid the occurrence of a reduction underneath an *act* operator.

Figures 6, 7 and 8 show a class of graphs: linear processes are encoded into an expression containing these graphs as constants, and parallel and sequential composition as operators. We assume a set

$$\{a, p\} \uplus \{X_a, X_p \mid X \in \mathcal{X}\}$$

that has no intersection with $\mathcal{N}$.

We write $0_{a,p}$ and $id_{a,p}$ as an abbreviation for $0_a \otimes 0_p$ and $id_a \otimes id_p$, respectively, and, similarly, we write 0_X and id_X as an abbreviation for $0_{X_a} \otimes 0_{X_p}$ and $id_{X_a} \otimes id_{X_p}$. Also, for a set of names Γ, we write 0_Γ and id_Γ as an abbreviation for $\bigotimes_{n\in\Gamma} 0_n$ and $\bigotimes_{n\in\Gamma} id_n$, respectively, and for a process P_ϵ, we write $id_{pv(P_\epsilon)}$ as an abbreviation for $\bigotimes_{X\in pv(P_\epsilon)} id_X$. The preceding expressions are well defined because the $\otimes$ operator is associative. The following definition introduces the encoding of extended processes (with no occurrence of name variables) into graphs with interfaces by mapping a process into a graph expression. We write $[\![M.P_\epsilon]\!]_\Gamma$ to denote 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 and $open_n$ graphs, respectively.

Definition 5.1 (encoding for processes). Let P_ϵ be a linear process with no occurrence of name variables and 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 Table 4.

Given a linear 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

$$(\{a, p\} \uplus \{X_a, X_p \mid X \in pv(P_\epsilon)\} \uplus \Gamma, \varnothing).$$

Its *enriched encoding* is the graph $[\![P_\epsilon]\!]_\Gamma \otimes go$, denoted by $[\![P_\epsilon]\!]_\Gamma^{go}$. Intuitively, it links a go edge to the $\diamond$ root node of each graph representing a process. This edge is needed for

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

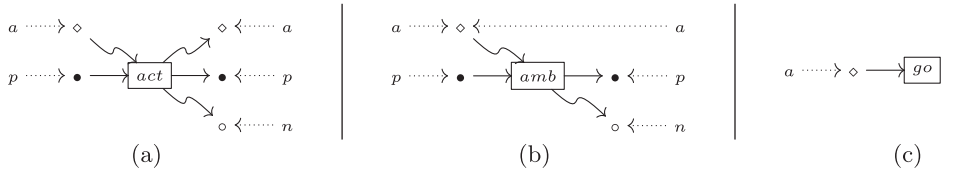


Fig. 6. Graphs (a) act_n (with $act \in \{in, out, open\}$), (b) amb_n and (c) go.

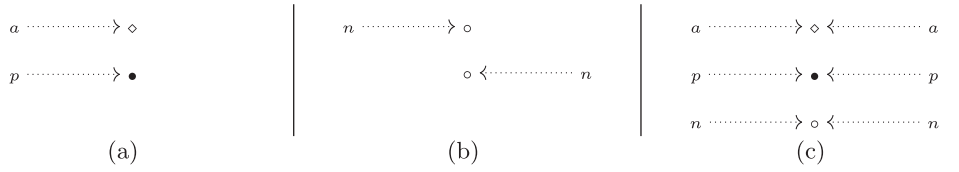


Fig. 7. Graphs (a) $\mathtt{0}_a$ and $\mathtt{0}_p$, (b) $\mathtt{0}_n$ and new_n, and (c) id_a, id_p and id_n.

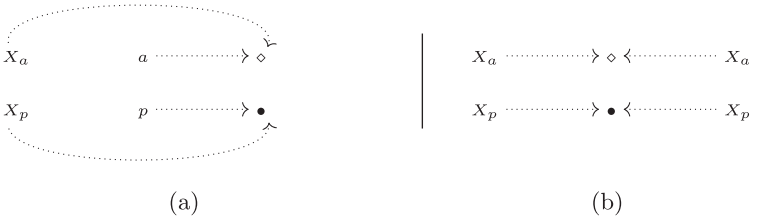


Fig. 8. Graphs (a) $\mathtt{0}_{X_a}$ and $\mathtt{0}_{X_p}$, and (b) id_{X_a} and id_{X_p}.

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

Table 4. *Encoding for pure processes.*

detecting the 'entry' point for the computation of the process. Its use will become clearer later on.

The encoding is fully abstract with respect to the structural congruence (Gadducci and Monreale 2008, Theorem 1).

Proposition 5.1. Let P and 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

$$[\![P]\!]^{go}_\Gamma = [\![Q]\!]^{go}_\Gamma.$$

Note that this result could be suitably extended to encompass linear processes.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

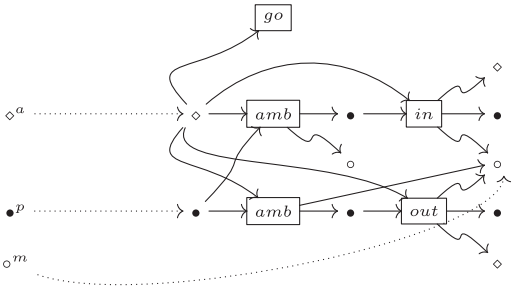


Fig. 9. Graph encoding for the process $(\nu n)(n[in\ m.0]|m[out\ m.0])$.

Example 5.1. Consider the pure process

$$Q = n[in\ m.0]|m[out\ m.0]$$

from Example 2.1. Figure 2 shows the encoding $[\![Q]\!]_{\{n,m\}}$ using Definition 5.1. The upper sub-graph in the encoding represents the left parallel component of Q, that is, the process $n[in\ m.0]$, and the lower sub-graph represents the right-hand component of the parallel operator, that is, $m[out\ m.0]$. The sub-graphs have the same roots, into which the nodes of the interface a and p are mapped, and they share the name node representing the ambient name m, which is used in both terms. This happens because the encoding is obtained *via* the $\otimes$-composition (Definition 3.3) of $[\![n[in\ m.0]]\!]_{\{n,m\}}$ and $[\![m[out\ m.0]]\!]_{\{n,m\}}$, shown in Figure 1 (left to right).

The enriched encoding $[\![P]\!]^{go}_{\{m\}}$ for pure process $P = (\nu n)Q$, also from Example 2.1, is shown in Figure 9. It is obtained from $[\![Q]\!]_{\{n,m\}}$ in two steps: the node n is first removed from the interface, giving $[\![P]\!]_{\{m\}}$, and the go edge is then attached to the activation node $\diamond$.

We will now consider the first step in more detail. By definition,

$$[\![P]\!]_{\{m\}} = (new_n \otimes id_m \otimes id_{a,p}) \circ [\![Q]\!]_{\{n,m\}}.$$

The graph with interface $(new_n \otimes id_m \otimes id_{a,p})$ has the same underlying graph of $id_{n,m,a,p}$, but the name n is missing from the input interface: its sequential composition with a graph having interface $\{n, m, a, p\}$ results in the same graph but without n among its inputs.

For the second step, note that the activation node $\diamond$ is linked to the edges representing the ambients n and m and to the capabilities $in\ m$ and $out\ m$: attaching the go edge to $\diamond$ will allow the ambients connected to it to be involved in a reduction.

6. Graph transformation for mobile ambients

In this section we present a GTS that models the MA reduction semantics. Figure 10 presents the rules of the GTS $\mathcal{R}_{amb}$, which simulates the reduction semantics $\rightarrow$ introduced in Section 2. The GTS $\mathcal{R}_{amb}$ only has three rules: p_{in}, p_{out} and p_{open}. These rules simulate the three axioms of the reduction relation: the rule p_{open} stands for the (OpenRed) axiom, and so on. Moreover, since we consider injective matches, we need an instance for the rules p_{in} and p_{out}, where the nodes labelled n and m may actually be coalesced, denoted p_{in-c} and p_{out-c}, respectively (these are not presented here).

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

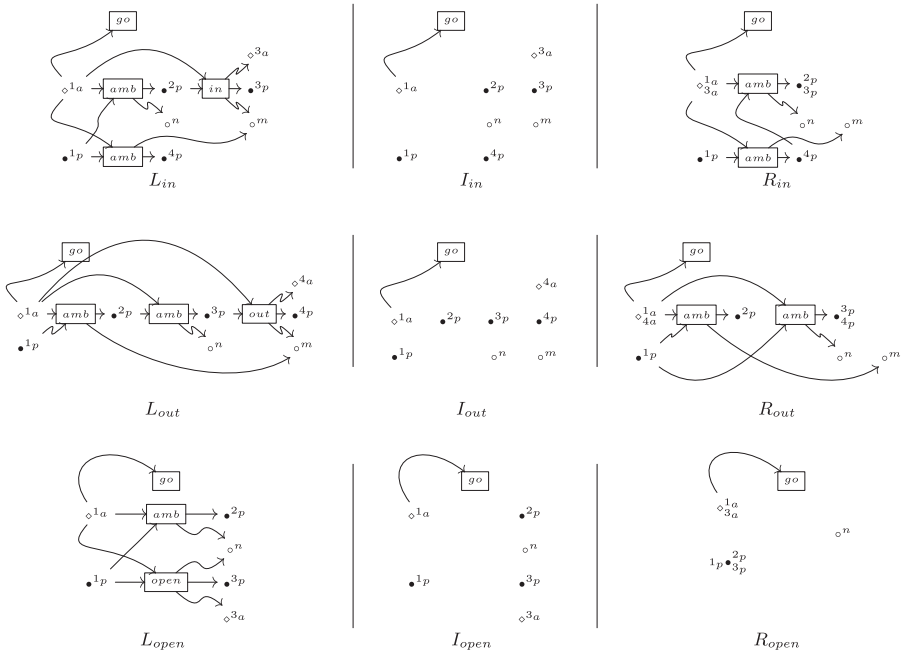


Fig. 10. The rewriting rules p_{in}, p_{out}, and p_{open}.

Node identifiers describe the rule action, and, of course, they are arbitrary: they correspond to the elements of the set of nodes and characterise the span of functions.

Three (plus two) rules suffice for recasting the MA reduction semantics. Indeed, the closure of reduction with respect to contexts is obtained by the embedding of a graph within a larger one. Moreover, no rule instance is needed: graph isomorphism takes care of structural congruence, and interfaces look after the renaming of free names.

Our encoding is fully abstract with respect to the reduction relation $\rightarrow$ (Gadducci and Monreale 2008, Theorem 2).

Proposition 6.1 (reductions versus rewrites). Let P be a pure process and Γ be a set of names such that $fn(P) \subseteq \Gamma$. If $P \rightarrow Q$, then $\mathcal{R}_{amb}$ entails a direct derivation

$$[\![P]\!]^{go}_{\Gamma} \Longrightarrow [\![Q]\!]^{go}_{\Gamma}.$$

Conversely, 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 correspondence holds since a rule is applied only if there is a match that covers a sub-graph with the go operator on top. This allows the occurrence of reductions inside activated ambients, but not inside capabilities.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

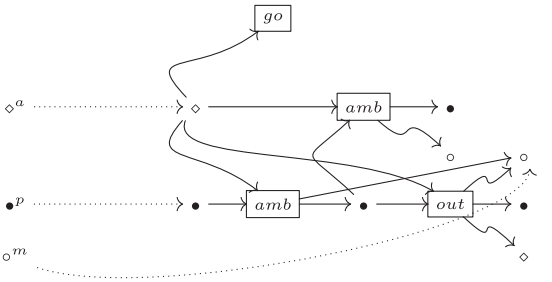


Fig. 11. Graph encoding for the process $(\nu n)(m[n[0]|out\ m.0])$.

Example 6.1. Consider the process

$$P = (\nu n)(n[in\ m.0]|m[out\ m.0])$$

from Example 2.1 again: its enriched graphical encoding is shown in Figure 9. The edge labelled *go* denotes the entry point for the computation of the process. All 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} rule to the graph in Figure 9 results in the graph in Figure 11, which encodes the process

$$(\nu n)(m[n[0]|out\ m.0]).$$

This rewriting step simulates the transition

$$P \to (\nu n)(m[n[0]|out\ m.0])$$

in Example 2.1. With respect to

$$Q = n[in\ m.0]|m[out\ m.0] \to m[n[0]|out\ m.0],$$

the application of the (ResRed) rule is immaterial: the occurrence of the restriction operator is mimicked by embedding L_{in} into a graph lacking n in the interface.

7. The synthesised transition system

In this section we apply the BC synthesis mechanism to $\mathcal{R}_{amb}$ to derive an LTS for graphs representing MA processes. We will begin in Section 7.1 by explaining the graphical counterpart of process variables, which we then use to present some rewriting steps using BCs in Section 7.2. In Section 7.4, we introduce a compact representation of the derived LTS by means of *minimal derivations*, which are extrapolated using two pruning techniques described in Section 7.3. We will use the resulting LTS in Section 8 to give a direct definition of a novel LTS for MA processes.

7.1. *Process variables, graphically*

In this section we will illustrate how a single BC transition may induce a reduction involving extended processes. To this end, consider the graph $J \rightarrowtail G$ in Figure 12 and the

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

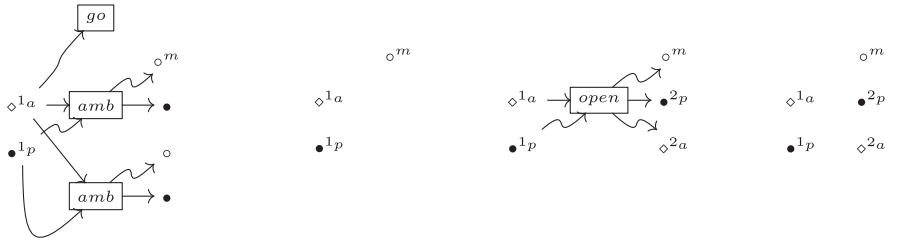


Fig. 12. The graphs with interfaces $J \rightarrowtail G$ and the context $J \rightarrowtail F \leftarrowtail K$.

diagram in Definition 4.3. The former represents the encoding of the process

$$S = (\nu n)(m[0] \mid n[0]).$$

The occurrence of nodes $\bullet^{1_p}$ and $\diamond^{1_a}$ in J ensures that the process represented by $J \rightarrowtail F$, $T = open\, m.0$, is put in parallel with S so that $J \rightarrowtail G^+$ corresponds intuitively 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 $K \rightarrowtail G^+$ represents $S \mid T_X$ for any process variable X and linear process $T_X = open\, m.X$.

Put differently, the context $J \rightarrowtail F \leftarrowtail K$ is the minimal context allowing the reaction since it could be further instantiated with any substitution of the process variable X.

7.2. *Examples of borrowed transitions*

In this section we show the application of the BC synthesis mechanism to the graphical encoding of a process. Consider the graph

$$J \rightarrowtail G = [\![P]\!]^{go}_{\{m\}},$$

where

$$P = (\nu n)(n[in\ m.0] | m[out\ m.0]).$$

In the following, we will discuss the possible transitions with source $J \rightarrowtail G$ induced by the rule

$$p_{in} : L_{in} \leftarrowtail I_{in} \rightarrow R_{in}$$

of $\mathcal{R}_{amb}$. Since a labelled transition might exist for each pair of monos $G \leftarrowtail D \rightarrowtail L_{in}$, in order to perform a complete analysis, we will need to consider all the pairs of monos $G \leftarrowtail D \rightarrowtail L_{in}$. However, we will only need to show some of the possible transitions generated by such pairs since, as we will see later, there is no need to check all the other pairs if we use the pruning techniques described in Section 7.3.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

7.2.1. *BC transition for D equal to L_{in}.*
When D is L_{in}, there is only one map into G. The resulting transition is shown in Figure 13. G^+ coincides with G, and C and H are constructed as in a standard DPO rewriting step. Since $J \rightarrowtail G$ needs no context for the reaction, the label of this transition is the identity context: two isomorphisms into the discrete graphs with three nodes $\{p, a, m\}$ (that is, it is the value of the expression $id_p \otimes id_a \otimes id_m$, see Section 5). Intuitively, this corresponds to an internal transition over processes, labelled with τ.

7.2.2. *BC transition for D equal to the upper sub-graph of L_{in}.*
When D is the sub-graph of L_{in} representing an ambient with a capability *in* inside it, there is again only one map into G. The resulting transition is shown in Figure 14. G^+ coincides with G in parallel with the graph representing an ambient m encoding the process

$$(\nu n)(n[in\ m.0]|m[out\ m.0]|m[X])$$

for some variable X. In order to reach G^+, we have $J \rightarrowtail G$ borrow the context $J \rightarrowtail F \leftarrowtail K$ (the syntactic context $-|m[X]$) from the environment. In the interface K, there is a node $\bullet^{4_p}$ pointing to the process node of F inside the ambient m, and this node represents a variable X, as described in Section 7.1. C and H are then constructed as in the standard DPO approach. Intuitively, $K \rightarrow H$ represents the process

$$m[out\ m.0]|m[n[0]|X],$$

where X is the variable occurring in the label $J \rightarrowtail F \leftarrowtail K$. This can be understood by observing that the node $\bullet^{4_p}$ of K points both to a node of H and to a node of F. Summarising, the ambient n moves into an ambient m provided by the environment.

7.2.3. *BC transition for D equal to the lower sub-graph of L_{in}.*
When D is the lower sub-graph of L_{in} consisting of the ambient edge alone, there are two possible maps into G: the map into the sub-graph of G representing the ambient m and the map into the sub-graph of G representing the restricted ambient n.

In the first case, we obtain the transition shown in Figure 15. G^+ coincides with G in parallel with the graph representing a fresh ambient name w having within it a capability *in m* encoding the process

$$(\nu n)(n[in\ m.0]|m[out\ m.0]|w[in\ m.X_2|X_1])$$

for some variables X_1, X_2. In order to reach G^+, we have $J \rightarrowtail G$ borrows the context $J \rightarrowtail F \leftarrowtail K$ (the syntactic context $-|w[in\ m.X_2|X_1]$) from the environment. As before, X_1 and X_2 are variables since, in the interface K, there are nodes $\bullet^{2_p}$ and $\bullet^{3_p}$. C and H are obtained by a standard DPO derivation. Intuitively, $K \rightarrow H$ represents the process

$$(\nu n)(n[in\ m.0]|m[out\ m.0|w[X_2|X_1]]).$$

Summarising, an ambient w from the environment enters the ambient m of a process P.

In the second case no transition is possible. G^+ coincides with G in parallel with a fresh ambient w having within it a capability *in n*, but the pushout complement of $J \rightarrowtail G \rightarrowtail G^+$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

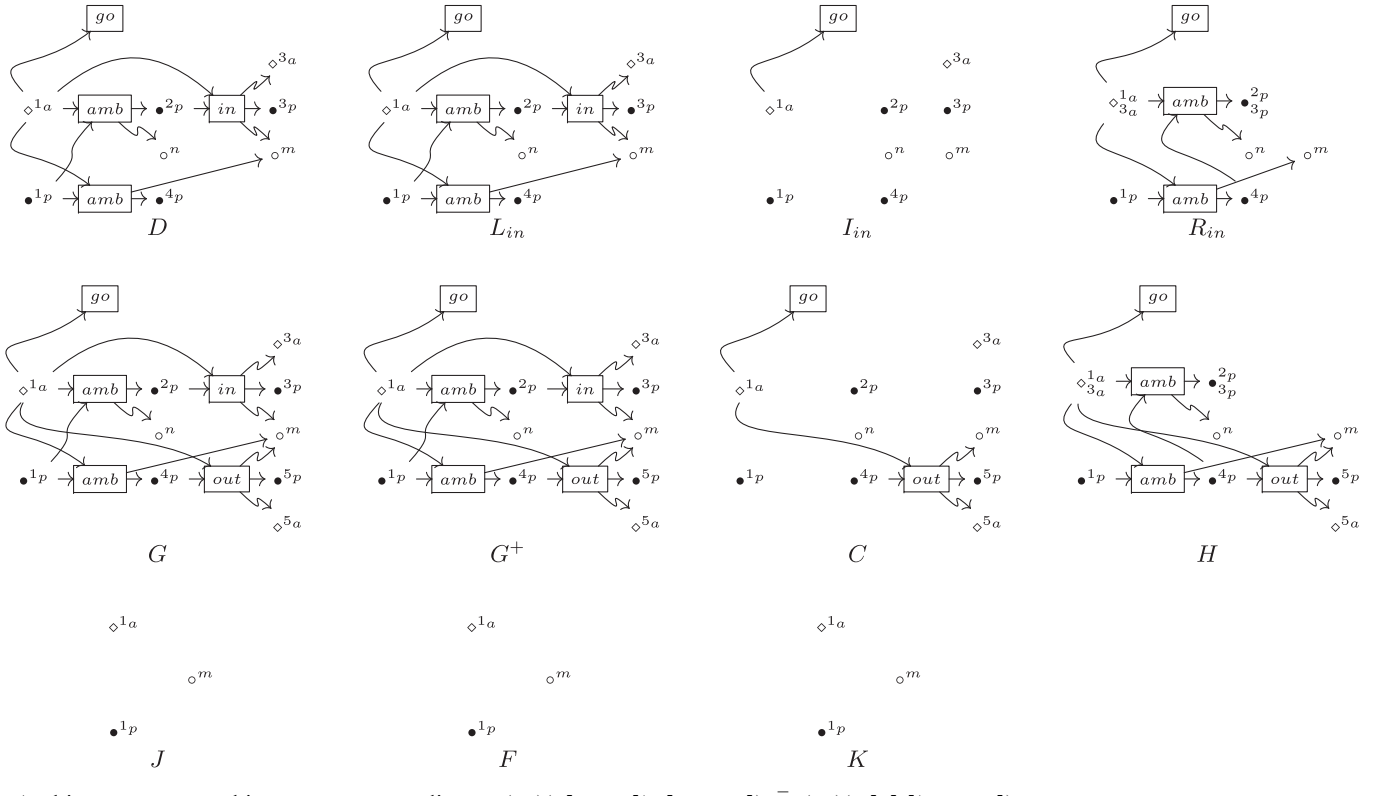


Fig. 13. Ambient n enters ambient m, corresponding to $(\nu n)(n[in\ m.0]|m[out\ m.0]) \xrightarrow{-} (\nu n)(m[n[0]|out\ m.0])$.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

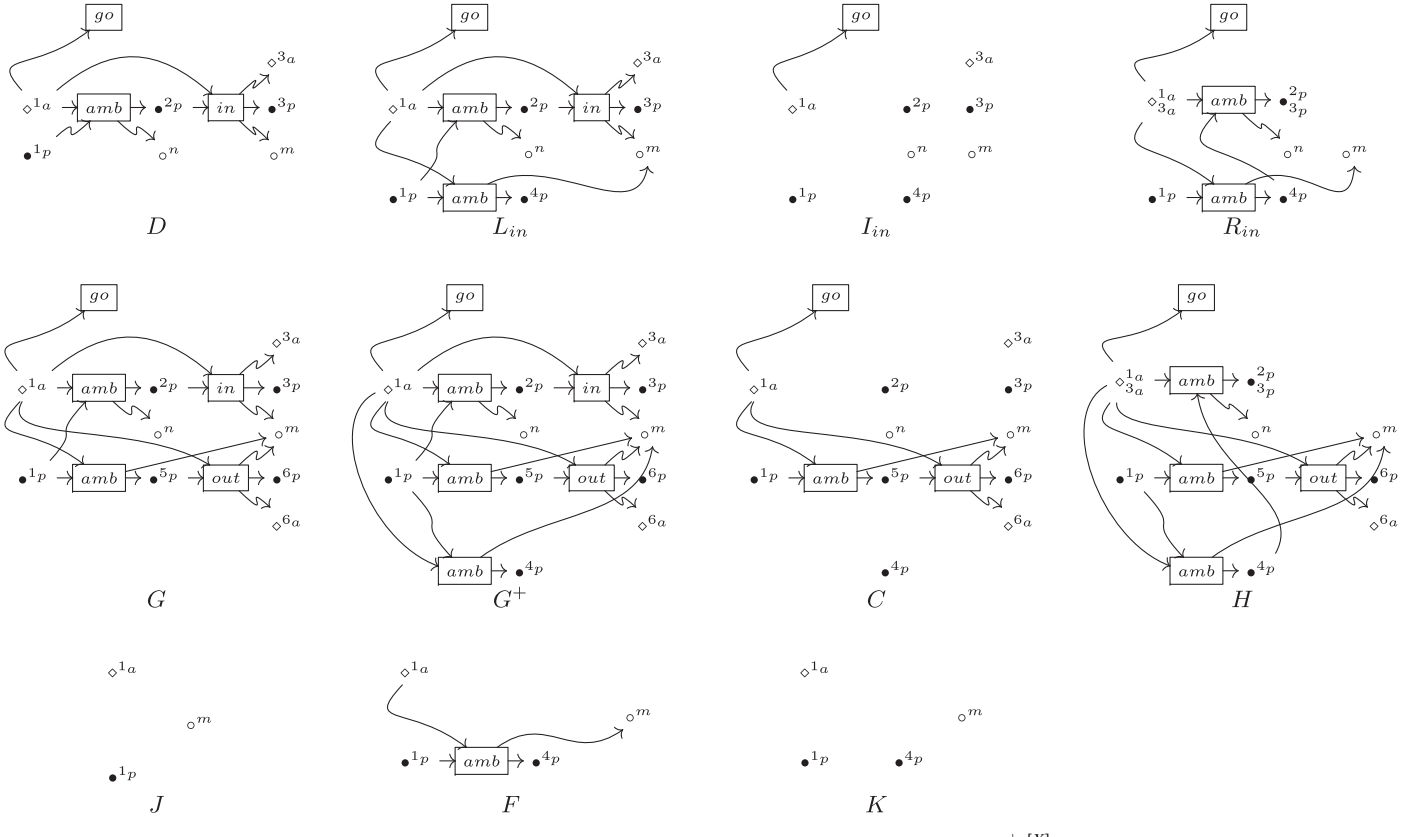


Fig. 14. Ambient n enters ambient m (from environment), corresponding to $(\nu n)(n[in\ m.0]|m[out\ m.0]) \xrightarrow{-|m[X]} (\nu n)(m[out\ m.0]|m[n[0]|X])$.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press



Fig. 15. Ambient w (from environment) enters ambient m, corresponding to $(\nu n)(n[in\ m.0]|m[out\ m.0]) \xrightarrow{-|w[in\ m.X_2|X_1]} (\nu n)(n[in\ m.0]|m[out\ m.0|\ w[X_2|X_1]])$.

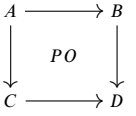
does not exist: n is restricted, so it does not belong to the interface J. Intuitively, no ambient from the environment can enter a restricted sibling ambient n.

7.3. *Reducing the borrowing*

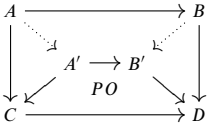
In order to derive the transitions originating from a graph $J \rightarrowtail G$, we need to analyse all the sub-graphs Ds of L_{in}, L_{out} and L_{open}. However, to shorten this procedure, we can use two pruning techniques, which were introduced in Bonchi *et al.* (2006a).

The first pruning technique exploits the observation that those items of L that are not in D must be glued to G through J. Consider a node n of D mapped to a node n' in L such that n' is the source or the target of an edge e not in D. Such graph items, which are called *boundary nodes*, are modelled using *initial pushouts* (Ehrig *et al.* 2006).

Definition 7.1 (initial pushout). Let the square



be a pushout. It is an *initial pushout* of $C \rightarrow D$ if, for every other pushout as in the diagram



(*)

there exist two unique morphisms $A \rightarrow A'$ and $B \rightarrow B'$ such that (*) commutes.

Since our chosen category of typed (hyper-)graphs has initial pushouts for all arrows (Ehrig *et al.* 2006), the previous discussion can be formalised by the following lemma (Bonchi *et al.* 2006a, Corollary 1).

Lemma 7.1. A graph with interfaces $J \rightarrow G$ 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 $D \rightarrowtail G$ and a morphism $J_D \rightarrow J$ such that square (1) in Figure 16 is an initial pushout and square (2) commutes.

This lemma allows us to prune the space of possible Ds dramatically. For instance, we can exclude those Ds having a continuation node (any process node represented by $\bullet$ that is not the root) as boundary node, since the only process node in the interface J is the root node. We can also exclude those Ds having as boundary node a name node that is not in the interface J, otherwise J_D should contain such a node and the morphism $J_D \rightarrow J$ would not exist. Consider, for example, the p_{open} rule. This means that when the node n is not in the interface J (the name n of the encoded process is bound), $J \rightarrow G$ can perform a BC rewriting step if the node n in D is not a boundary node, that is, if D contains both the *open* and the *amb* edges.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

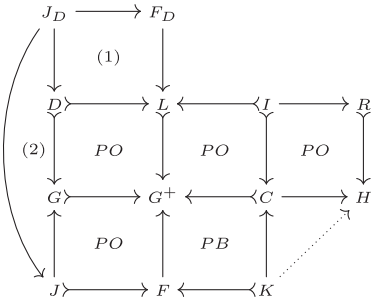


Fig. 16. BC construction with commuting squares (1) (the initial pushout of $D \rightarrowtail L$) and (2).

A further pruning, which is partially based on proof techniques presented in Ehrig and König (2006), is performed by excluding those Ds that generate a BC transition that is not relevant for bisimilarity. We may, for example, exclude those Ds that only contain nodes since they can be embedded in every graph (with the same interface) generating the same transitions. For our case study, those transitions generated by a D having the root node without the *go* edge are irrelevant since 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 that the resulting states of these two transitions only differ in the number of *go* edges attached to the root: the state resulting after the first transition has two *go*'s, but the state resulting after the second transition has only one. These states are bisimilar since the number of *go*'s does not change the behaviour (Bonchi *et al.* 2006a, Lemma 12).

The two pruning techniques presented in this section mean we only need to consider the partial matches D shown in Figures 18, 19 and 20, together with those Ds obtained from those in Figures 19 and 20 by coalescing the name nodes n and m.

7.4. *Minimal transitions*

In Section 7.3 we heavily pruned the space of all possible Ds using Lemma 7.1 and by excluding those Ds that generate a BC transition that is irrelevant for bisimilarity. However, reasoning with the synthesised LTS is still hard (this is usually with derived LTSs, as pointed out in Baldan *et al.* (2006) and Bonchi (2008), where an SOS presentation of the synthesised LTS was deemed to 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) synthesised LTS.

Inspired by Lemma 7.1, which provides necessary and sufficient conditions for performing a transition, we consider the graphs $J_D \rightarrow D$ for all those Ds that have not been pruned in Section 7.3 and for J_D containing only the boundary nodes of D .

The minimal transitions have the shape shown in Figure 14, where the leftmost square in the lower row is an initial pushout. Figures 18, 19 and 20 represent these transitions concisely by showing their starting graph D, the label $J_D \rightarrowtail F_D \leftarrowtail K_D$ and the resulting graph R. These three figures represent the minimal transitions generated by the rules p_{open}, p_{in} and p_{out}. The minimal transitions generated by the rules p_{in-c} and p_{out-c} should also

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

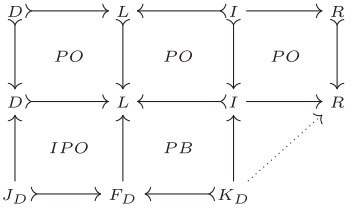


Fig. 17. Transition shape.

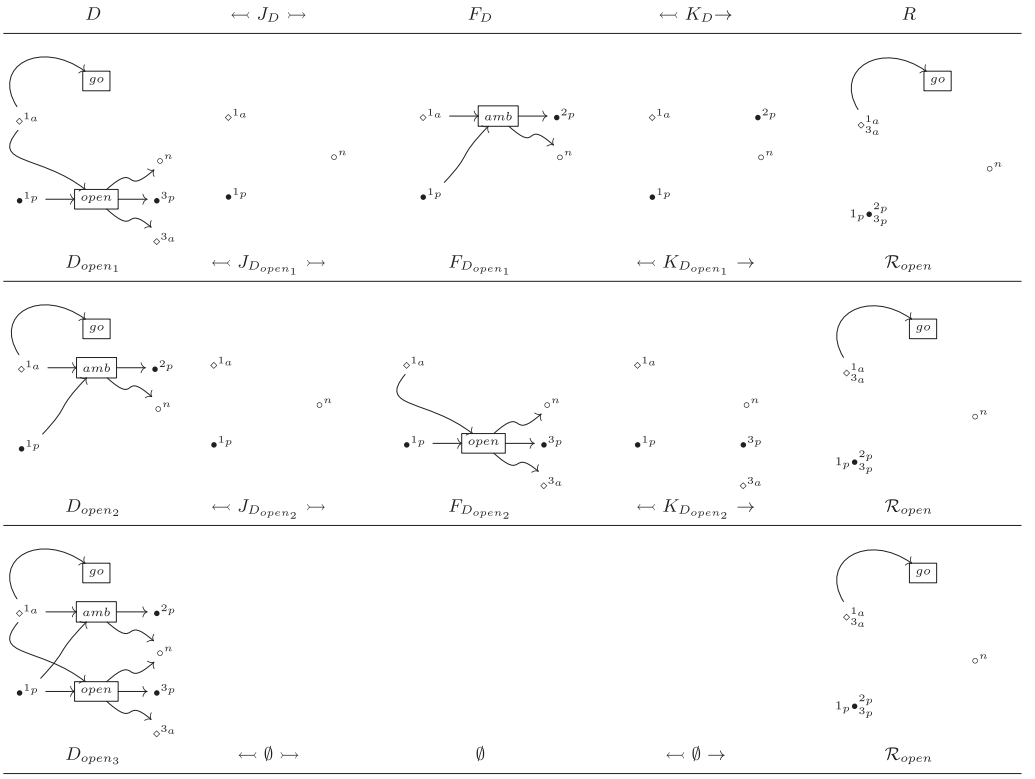


Fig. 18. The minimal transitions generated by the rule p_{open}.

be considered, but they are easily described from those of p_{in} and p_{out}. More precisely, for each minimal transition with D_{in_x}, there exists a minimal transition generated by p_{in-c}, where the relevant graphs are obtained by coalescing the nodes n and m (and similarly for D_{out_x} and p_{out-c})[†].

All the transitions originating from a process encoding $J \rightarrowtail G$ can be characterised using these minimal transitions. By Lemma 7.1, 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

[†] Note also that we do not need to consider the coalesced version for the rule with D'_{in_i} either, since it would coincide with the minimal transition for D_{in-c_i}, for all i.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

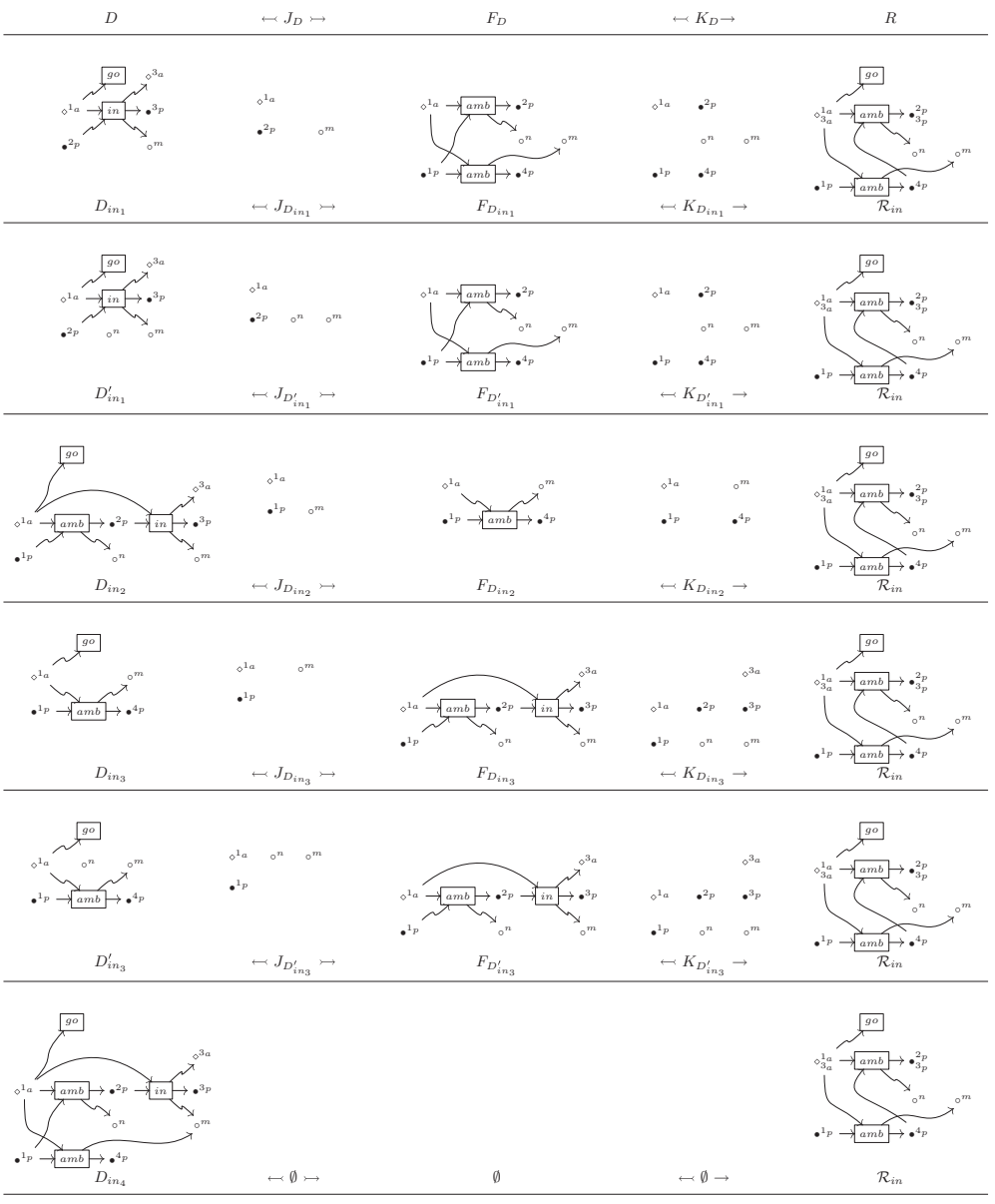


Fig. 19. The minimal transitions generated by the rule p_{in}.

the minimal transitions, and a morphism $J_D \to J$ such that the square (2) in Figure 16 commutes.

The label of the rewriting step can be obtained from the label of the minimal transition. First 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 modelled process (as expected by our encoding); and the root nodes of the graph D when they are not in J_D. So the graph F only contains the whole graph F_D and all the nodes of J. Indeed, as shown

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

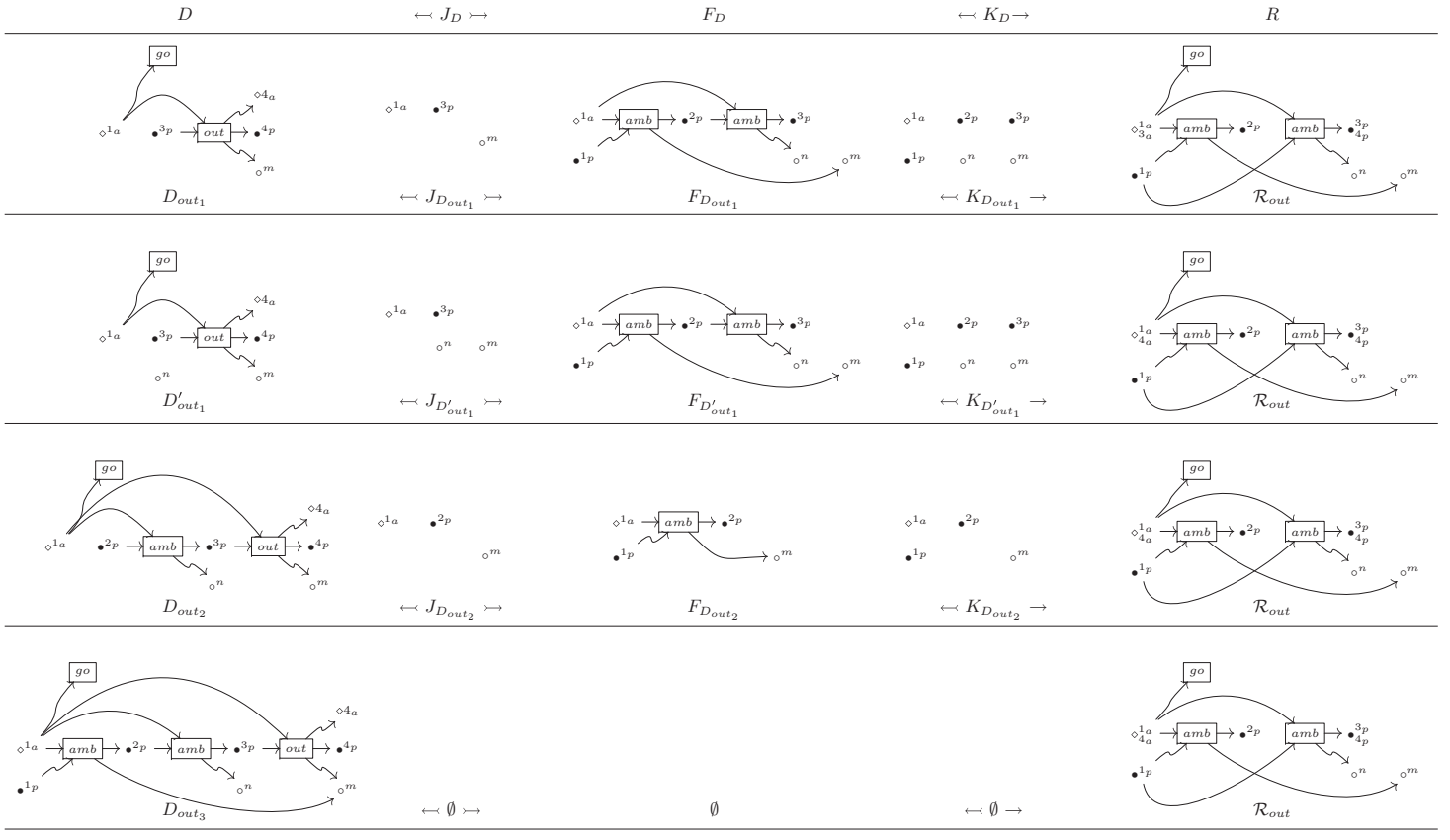
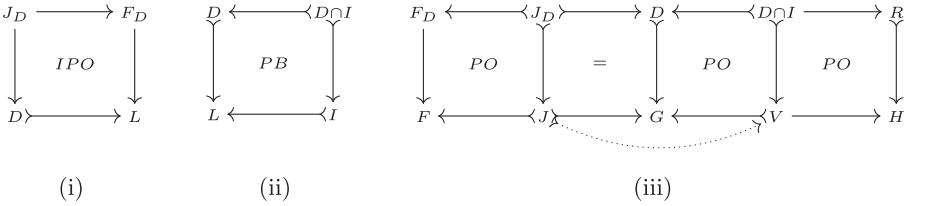


Fig. 20. The minimal transitions generated by the rule p_{out}.



(i) (ii) (iii)

Fig. 21. Diagrams used in Proposition 7.1.

in the proposition below, which is adapted from Bonchi *et al.* (2006a, Proposition 4), F can be obtained as the pushout of $J_D \to F_D$ and $J_D \to J$.

Proposition 7.1. Let:

— $p : L \leftarrowtail I \to R$ be a production of $\mathcal{R}_{amb}$;
— $J \rightarrowtail G$ be a graph with interfaces; and
— $d : D \rightarrowtail L$ be a mono such that square (i) in Figure 21 is the initial pushout of d and square (ii) is a pullback.

Then there is 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 are a mono $D \rightarrowtail G$, a graph V and a morphism $J_D \to J$ such that diagram (iii) in Figure 21 commutes and F and H are built as shown there.

It is easy to prove that K is a discrete graph containing exactly the nodes of F (that is, K consists only of the nodes of J and K_D).

Finally, the resulting graph H is obtained by replacing the sub-graph D in G with R. As shown in Proposition 7.1, this 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 $I \rightarrowtail L$.

As an example, consider the BC rewriting step in Figure 13. This is derivable by the minimal transition for D_{in_4} (shown in Figure 19). We first note that there exist $D_{in_4} \to G$ and $\varnothing \to J$ such that the square (2) in Figure 16 commutes. Also, F is equal to J since it consists of the composition of $F_{D_{in_4}}$ (that is, $\varnothing$) and J. The new interface K is equal to F since it contains all and only the nodes of J and $K_{D_{in_4}}$ (that is, $\varnothing$). The arriving state H is obtained simply by replacing D_{in_4} with R_{in}.

8. A new LTS for mobile ambients

In this section we present the LTS $\mathcal{D}$ defined directly over MA processes. The inference rules describing this LTS are obtained from the transitions of the LTS on graphs presented in Section 7.4. The labels of the transitions are unary contexts, that is, terms of the extended syntax with a hole '—'. The formal definition of our LTS is given in Figures 22 and 23.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

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

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

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

Fig. 22. The internal transitions of the LTS $\mathcal{D}$ (for $\mathcal{C}[-]$ an enabling context).

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

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

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

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

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

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

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

Fig. 23. The environmental transitions of the LTS $\mathcal{D}$.

8.1. *The labelled rules on processes...*

The rules in Figure 22 represent the τ-actions modelling internal computations. The labels of the transitions are contexts composed purely of a hole $-$, while the resulting states are pure processes. Also, an *enabling* context may only contain ambients and parallel operators. The rule InTau enables an ambient n to enter a sibling ambient m. The rule OutTau enables an ambient n to leave its parent ambient m. Finally, the rule OpenTau models the opening of an ambient n. These three rules exactly derive the MA reduction relation, so they could be replaced by the rules in Table 10.

The rules in Figure 23 model the interactions of a process with its environment, and the labels and resulting states contain process and name variables. We define the LTS $\mathcal{D}_{\mathcal{I}}$ for MA pure processes by instantiating all these variables.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Definition 8.1. Let P and Q be pure processes and $C[-]$ be a pure context. We have a transition

$$P \xrightarrow{C[-]}_{\mathcal{D}_{\mathcal{I}}} Q$$

if there exist a transition

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{D}} Q_\epsilon$$

and a substitution σ such that

$$Q_\epsilon \sigma \equiv Q$$

and

$$C_\epsilon[-]\sigma = C[-].$$

Recall that substitutions map name variables into ambient names and process variables into pure processes, and that they can never capture bound names.

The rule Open models the opening of an ambient provided by the environment. More explicitly, 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 only possible by 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.

By contrast, the rule CoOpen models an environment that opens an ambient of the process itself. 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 ambients are enabled. In the rule CoIn, an ambient provided by the environment enters an ambient of the process. In the rule OutAmb, an ambient of the process exits from an ambient provided by the environment, while in the rule Out both ambients exit.

The LTS $\mathcal{D}$ does not conform to the SOS style since the premises of the inference rules are just constraints over the process structure. This is because the rules of the LTS $\mathcal{D}$ are derived from the minimal transitions. Each rule corresponds to one of the minimal transitions presented in Section 7.4 and is obtained as described below.

8.2. *...from the borrowed rules on graphs*

Note 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 exist a mono $D \rightarrowtail G$ for some D of a minimal transition and a morphism $J_D \to J$ such that the square (2) in Figure 16 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. Hence, we obtain an inference rule for each minimal transition: 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

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

of the borrowed transition. 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 will look closely at the correspondence between the rule OPEN and the first minimal transition in Figure 18.

Consider a graph $J \rightarrowtail G$ representing the encoding for a process P. If there exist a mono $D_{open_1} \rightarrowtail G$ and a morphism $J_{D_{open_1}} \rightarrow J$ such that the square (2) in Figure 16 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 consist of $J_{D_{open_1}}$, $F_{D_{open_1}}$ and $K_{D_{open_1}}$, respectively, together with the free names of P. We 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 16 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, though it may possibly be 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 $J_{D_{open_1}}$, that is, 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 $K_{D_{open_1}}$, that is, 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 $(\nu A)(P_1|X_1|P_2)$ corresponds exactly to the state H from the BC transition. Indeed, in the interface K of the graph $K \rightarrow H$ the node modelling the process variable X_1 also occurs, and this represents a process provided by the environment.

Note that while there are 13 minimal transitions, there are only 10 rules in Figures 22 and 23. This is because 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}, COIN by D_{in_3} and D'_{in_3}, and OUT by D_{out_1} and D'_{out_1}. We will just show the last of these since the others are analogous.

In the minimal transition with D_{out_1}, two ambients are borrowed from the environment. The first has the name m (that is, 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 $K_{D_{out_1}}$). So this

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

transition 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 occurs in $J_{D'_{out_1}}$, it should appear in the interface J, that is, it must be free, so 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, so we can put their premises together and represent them compactly using the rule Out of Figure 23. Substituting the name n with a name variable x basically guarantees that any actual name can be substituted for n, even m (thanks to D_{out-c_1}), as long as it does not occur in A.

9. An SOS presentation for the derived LTS $\mathcal{D}$

In the previous two sections we described a semi-automatic methodology for distilling an LTS $\mathcal{D}$. In this section we introduce a set of SOS rules, tailored over $\mathcal{D}$, such that the associated LTS $\mathcal{S}$ coincides with the LTS $\mathcal{D}$. The rules for $\mathcal{S}$ are shown in Figure 24, where we omit the symmetric rules for the parallel operator.

The rules in the first two rows of Figure 24 model internal computations, and are obtained from the rules in Figure 22. Since these rules derive the same transition relation of the reduction relation over MAs, we replace them with the reduction rules labelled with the identity context $-$. Hence, we obtain the axioms modelling the execution of the capabilities of the calculus and a structural rule for each ambient, parallel and restriction operators. Note that the use of the structural congruence in the premise of the rules InTau, OutTau and OpenTau is not standard. Nevertheless, it allows a concise presentation of the LTS and, more importantly, it makes it easier to compare with the proposal of Rathke and Sobociński (2008b), where the authors assume the implicit presence of a rule closing the LTS under the structural congruence. However, we will also give a standard SOS presentation in Appendix C.

The remaining rules in Figure 24, which model the interactions of a process with its environment, are obtained from the rules in Figure 23. More explicitly, we derive three rules for each of the rules in Figure 23. First, we determine the axiom by considering the minimal process needed by the reduction for it to occur. For example, with respect to the rule In of the LTS $\mathcal{D}$, the minimal process allowing the reduction is $in\ m.P_1$, and we thus 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 relative structural rules in SOS style. Once again, for the rule In of the LTS $\mathcal{D}$, if

$$P \xrightarrow{x[-|X_1]|m[X_2]} P\epsilon,$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fig. 24. The LTS $\mathcal{S}$.

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, we have the transition

$$P|Q \xrightarrow{x[-|X_1]|m[X_2]} P_\epsilon \left\{ {}^{Q|X_1}/_{X_1} \right\}.$$

However, if

$$P \xrightarrow{x[-|X_1]|m[X_2]} P\epsilon$$

and $m \neq a$, then we have the transition

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

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

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

9.1. *Deriving axioms*

As explained in Section 7.4, a minimal transition represents a BC transition in which the starting graph is the smallest graph allowing a BC rewriting with respect to a given rule and a given partial match. The graph D of a minimal transition thus represents the minimal process needed for the reduction modelled by the BC transition to occur. This means that each minimal transition represents an axiom of the SOS LTS.

As an example, consider the minimal transition for D_{in_1}. The graphs D_{in_1} represents the process *in* $m.0$, 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 7.4, starting from the label of the BC transition, we obtain the label of the process transition, which in this case is $x[-|X_1]|m[X_2]$, with x a name variable. The resulting process is instead represented by the graph R_{in}, which 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, consider the similar 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 24 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 24.

9.2. *Deriving rules for the parallel operator*

The structural rules can be obtained by analysing the interface J_D of the minimal transition, whose nodes represent the 'handles' of D for interacting with the environment. Since J_D in a minimal transition contains the root nodes of D, we can add a graph I representing a process Q in parallel with D to give a graph $J \rightarrowtail G$ where J consists of J_D plus the free names of Q.

Now, since there are a mono $D \rightarrowtail G$ (since G consists of the graph D in parallel with I) and a morphism $J_D \to J$ such that the square (2) in Figure 16 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 plus the free names of Q. In terms of processes, this means that if

$$P \xrightarrow{C[-]} P\epsilon,$$

then we also have a transition labelled $C[-]$ originating from $P|Q$.

We now consider the minimal transition for D_{in_1} again. By analysing the interface $J_{D_{in_1}}$, we can get the structural rule for the parallel operator. Since $J_{D_{in_1}}$ contains the root nodes of D, we can add a graph I representing a process Q in parallel with D to give a graph

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

$J \rightarrowtail G$ where J consists of $J_{D_{in_1}}$ plus 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 consist of $F_{D_{in_1}}$ and $K_{D_{in_1}}$ plus 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 we also have a transition labelled with $x[-|X_1]|m[X_2]$ originating from $P|Q$. The resulting process is represented by the graph H, which is obtained by replacing D_{in_1} with R_{in} in G. Note that the node $\bullet_{2_p}$ after the reduction is under the ambient x and, moreover, it represents a process variable in P_ϵ. This means that the graph modelling Q (which has as root the node $\bullet_{2_p}$) after the reduction is under the ambient x and it is in parallel with the process variable X_1. Therefore the resulting process is

$$P_\epsilon \left\{ {}^{Q|X_1}/_{X_1} \right\}.$$

9.2.1. *Deriving rules for the restriction operator.*
The structural rules for the restriction operator can also be obtained by analysing 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 $J_D \rightarrow J$ such that the square (2) in Figure 16 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 $J_D \rightarrow J'$ such that the square (2) in Figure 16 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 $(va)P$, there is also a transition labelled $C[-]$ if the names belonging to J_D do not belong to a.

On the basis of these remarks, 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 there is a transition for the process $(va)P$ with the same label leading to the process $(va)P_\epsilon$ if the name m (which 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. However, 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 $J_D \rightarrow J$ such that square (2) in Figure 16 commutes.

9.2.2. *Equivalence between the LTSs.*
In the same way as for $\mathcal{D}$, we will define the LTS $\mathcal{S}_\mathcal{I}$ for pure processes for $\mathcal{S}$ by instantiating all the variables of the labels and the resulting states.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Definition 9.1. Let P and Q be pure processes and $C[-]$ be a pure context. Then we have a transition

$$P \xrightarrow{C[-]}_{\mathcal{S}_\mathcal{I}} Q$$

if there exist a transition

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

and a substitution σ such that

$$Q_\epsilon \sigma \equiv Q$$

and

$$C_\epsilon[-]\sigma = C[-].$$

As stated below, the LTSs $\mathcal{S}_I$ and $\mathcal{D}_I$ coincide (the proof is given in Appendix A).

Theorem 9.1. Let P be a pure process and $C[-]$ be a pure context. Then,

$$P \xrightarrow{C[-]}_{\mathcal{D}_\mathcal{I}} Q$$

if and only if

$$P \xrightarrow{C[-]}_{\mathcal{S}_\mathcal{I}} Q.$$

10. Equivalence between LTSs

In this section we show the equivalence between our LTS $\mathcal{S}_\mathcal{I}$ defined on pure processes and the LTS proposed in Rathke and Sobociński (2008b, Figures 6, 7 and 8).

Their LTS, which is closed under structural congruence, is split 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 LTS $\mathcal{C}$ and $\vec{M}$ by LTS $\mathcal{A}$. If

$$P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} Q,$$

then α identifies the minimal context needed by the pure process P to react, and $\vec{M}$ is a list of pure processes and ambient names instantiating the context components.

The first column of Table 5 shows the labels α of the LTS $\mathcal{C}$, while the second column presents the context χ_α that each label identifies (the correspondence is formally explained in Rathke and Sobociński (2008b, Lemma 6)). Note that each context χ_α contains a set of typed numbered holes: those of type N can be instantiated with ambient names, and those of type Pr with pure processes. Thus, in a transition

$$P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} P',$$

the tuple $\vec{M}$ instantiates all context components, that is, each hole of χ_α of process type Pr (different from 1_{Pr}) and name type N. The hole 1_{Pr} represents the hole that has to be instantiated with the process P, and it corresponds to the hole denoted $-$ in our contexts.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

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

Table 5. *The correspondence between* α, χ_α, $C^\alpha_\epsilon[-]$, $\vec{M}^\alpha_\sigma$ *and* σ^α_M.

For example, if

$$P \xrightarrow{in\,m\downarrow\vec{M}}_{\mathcal{CA}} P',$$

then the tuple $\vec{M}$ instantiates the holes 2_{Pr}, 3_N and 4_{Pr} of the context $\chi_{in\,m}$, that is, it has the shape $\vec{M} : Q, n, R$ for Q and R pure processes and n an ambient name.

We can see immediately 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 5. This correspondence is also shown in the third column of the table, where $C^\alpha_\epsilon[-]$ denotes the label of our LTS $\mathcal{S}$ corresponding to the context χ_α in the second column.

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 the name and process variables of $C^\alpha_\epsilon[-]$ and the holes of χ_α of type N and Pr, respectively. Consider, for example, the label

$$C^{in\,m}_\epsilon[-] = x[-|X_1]|m[X_2]$$

and the context

$$\chi_{in\,m} = 3_N[1_{Pr}|2_{Pr}]|m[4_{Pr}].$$

They both have the same shape, and 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}.

As explained in Section 9, a substitution σ for a context $C_\epsilon[-]$ provides an instantiation for its variables. For instance, a substitution for

$$C^{in\,m}_\epsilon[-] = x[-|X_1]|m[X_2]$$

must have the shape

$$\{^Q/_{X_1}, ^n/_x, ^R/_{X_2}\}$$

for Q and R pure processes and n an ambient name. Since there is a correspondence between the holes of a context χ_α of type N and Pr and the name and process variables of the context $C^\alpha_\epsilon[-]$, any tuple $\vec{M}$ for χ_α determines a unique substitution σ_M for $C^\alpha_\epsilon[-]$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

instantiating each variable with the value used by $\vec{M}$ to instantiate the hole corresponding to that variable. Analogously, a substitution σ for $C^{\alpha}_{\epsilon}[-]$ determines a unique 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 for the holes 2_{Pr}, 3_N and 4_{Pr}. The substitution σ_M (induced by $\vec{M}$) for the context

$$C^{in\,m}_{\epsilon}[-] = x[-|X_1]|m[X_2]$$

is

$$\left\{ {}^{Q}/_{X_1}, {}^{n}/_{x}, {}^{R}/_{X_2} \right\}.$$

Analogously, it is possible to determine the tuple $\vec{M}$ from the substitution σ_M. The final two columns of Table 5 show the shape of the tuples $\vec{M}^{\alpha}_{\sigma}$ and σ^{α}_{M} for each α.

Proposition 10.1. 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 10.2. Let P be a pure process and 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

$$P \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} Q.$$

The main result of this section now follows from the two propositions above (see Appendix B for the proofs) and the definition of the LTS $\mathcal{S}_{\mathcal{I}}$ (Definition 9.1).

Theorem 10.1. 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

$$P \xrightarrow{C^{\alpha}_{\epsilon}[-]\sigma}_{\mathcal{S}_{\mathcal{I}}} Q.$$

Conversely, if

$$P \xrightarrow{C[-]}_{\mathcal{S}_{\mathcal{I}}} Q,$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

then there are α and a unique (up-to $\equiv$) tuple $\vec{M}$ such that

$$C[-] = C^{\alpha}[-]$$

and

$$P \xrightarrow{\alpha\downarrow\vec{M}}_{\mathcal{CA}} Q.$$

11. Conclusions and related and future work

In this paper we have exploited a graphical encoding for MAs (Gadducci and Monreale 2008) to distil an LTS on (processes encoded as) graphs. This LTS can be obtained semi-automatically by applying the BC technique to the GTS associated with the calculus after using two pruning techniques to remove some reductions. The LTS defined on graphs is then used to infer a novel LTS $\mathcal{D}$ directly defined on MA processes. We have also presented a set of SOS rules for MAs showing that the LTS $\mathcal{S}_I$ they induce on pure processes coincides with $\mathcal{D}_I$. Finally, we have proved that our $\mathcal{S}_I$ is equivalent to an alternative proposal in Rathke and Sobociński (2008b).

For simplicity, we have only considered the finite, communication-free fragment of MAs. However, a graphical encoding for the whole calculus could be obtained along the lines of the solution in Bonchi *et al.* (2006a), in which case, the technique presented in this paper could also be applied to obtain an LTS for it.

In spite of the interest in MAs, there has been little work addressing their labelled semantics. After the early attempts in Gordon and Cardelli (2003) and (*via* a graphical encoding) Ferrari *et al.* (2001), we are only aware of the papers by Merro and Zappa-Nardelli, collected together in Merro and Zappa Nardelli (2005), and Rathke and Sobociński (2008b; 2010).

We have already discussed the LTS introduced in Rathke and Sobociński (2008b), but note further that they also employ a general systematic procedure for deriving LTSs that they had previously introduced in Rathke and Sobociński (2008a), even if our use of the BC technique automatically guarantees that the strong bisimilarity on the derived LTS is a congruence. Rathke and Sobocinski (2010), unlike Rathke and Sobociński (2008b), presents a purely SOS presentation for the LTS and avoids the use of the structural congruence in deriving the transition relation. The solution boils down to adding some symmetric counterparts to the previous set of rules to deal with the commutativity of the parallel operator – for example, compare Rathke and Sobocinski (2010, Figure 2) with Rathke and Sobociński (2008b, Figure 2). The same technique can be pursued in our approach, but at the expense of a more complex set of inference rules, as shown in Appendix C. Note that the key Theorem 10.1, stating the correspondence between our LTS and Rathke and Sobociński's LTS for MAs, would still hold in the novel setting.

The LTS proposed in Merro and Zappa Nardelli (2005) is restricted to *systems*, which are processes obtained by the parallel composition of ambients. For this reason, our rules In, Open and Out have no counterpart in their work. Instead, the rules InAmb, CoIn and OutAmb correspond exactly to the rules (Enter), (Co-Enter) and (Exit) in Merro and Zappa Nardelli (2005, Table 6). Moreover, our rule CoOpen corresponds roughly to

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

their (Open). Indeed, the former inserts a process into the context $-|open\ n.X_1$, while the latter inserts one into $k[-|open\ n.X_1|X_2]$ (also due to their restriction to systems). Unlike our LTS, their labels for the rules (Enter) and (Exit) contain the name of the migrating ambient n, thus requiring two extra rules (Enter Shh) and (Exit Shh) to deal with the possible restriction of n.

For the practitioner, the main interest of our work lies in the presentation of a succinct LTS for MAs, and the associated set of SOS rules. However, we believe that our work represents a relevant case study for the theory of reactive systems (Leifer and Milner 2000). As pointed out in the introduction, BC rewriting and bigraphical reactive systems (Milner 2006) are both instances of this theory. Together with Bonchi *et al.* (2006a), the current paper shows that the BCs approach is quite effective in deriving LTSs for process calculi, and it seems to confirm the advantage of BCs over graphs with interfaces with respect to bigraphs. In bigraphs, all reduction rules must be ground, so the labels and the arriving states of the derived transitions are also ground. Instead, rewriting using BCs allows us to employ a few non-ground rules (as shown in this paper), and the resulting transitions have labels and arriving states containing (process and name) variables. This feature was irrelevant for calculi such as CCS and π since the variables in the labels occur 'outside' the arriving state and can be forgotten. Consider, for example, the CCS transition

$$a.b \xrightarrow{-|\bar{a}.Y} b|Y$$

derived from the (non-ground) rule $a.X|\bar{a}.Y \longrightarrow X|Y$. The behaviour of the process $b|Y$ is equivalent to b: their interaction is basically restricted to processes offering a $\bar{b}$ action, and we can thus avoid considering Y. In the case of MAs, accounting for non-ground states is fundamental because process variables may occur nested inside ambients in arriving states.

Moreover, although the main reason for introducing the SOS LTS was to make it easier to compare our results with the LTS proposed in Rathke and Sobociński (2008b), its derivation could also represent a first step towards solving one of the problems of the theory of reactive systems, which concerns the flat structure of the distilled LTSs. Indeed, as discussed in Section 9, the SOS rules have not been distilled by hand, but through an analysis of the minimal transitions, which could suggest an automatic procedure for deriving SOS LTSs for reactive systems.

However, the relevance of this work for the theory of reactive systems is not limited to the above observations. The first author has shown in Bonchi (2008) that in reactive systems, the bisimilarity on the derived LTS is usually too strict, and *saturated bisimilarity* (that is, the bisimilarity over the LTS having all contexts as labels) is often more adequate, such as for logic programming, open π-calculus (Bonchi *et al.* 2006b) and Petri nets (Bonchi and Montanari 2007). The present work provides a further successful test of the above claim. In fact, the standard notion of bisimilarity over our LTS is too strict since it allows us to observe the ability of an ambient to migrate, which should be unobservable, as pointed out in Merro and Zappa Nardelli (2005). For this reason, Rathke and Sobociński added two extra rules to their LTS, while Merro and Zappa Nardelli chose an asymmetric definition of bisimilarity. The latter solution is reminiscent of *semi-saturated bisimulation*

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

(Bonchi *et al.* 2006b) – instead of requiring that two bisimilar processes must perform transitions with the same label, semi-saturated bisimulation requires that

$$\text{if } P \xrightarrow{C[-]} P_1, \text{ then } C[Q] \text{ reduces to } Q_1 \text{ and } P_1 \; R \; Q_1.$$

It is worth noting that the second and third points of Merro and Zappa Nardelli (2005, Definition 3.2) have this shape (the labels $*.enter_n$ and $*.exit_n$ are related to our contexts $-|n[X_1]$ and $n[-|X_1]$). We have made this correspondence precise by introducing the notion of a *barbed* bisimilarity for reactive systems (Bonchi *et al.* 2009c), and applying it to MAs. We have also further sketched a correspondence between barbs and semi-saturation (Bonchi *et al.* 2009b): the detailed presentation of the framework obtained by combining the two proposals is going to appear elsewhere.

Appendix A. Equivalence between the LTS $\mathcal{D}_{\mathcal{I}}$ and the LTS $\mathcal{S}_{\mathcal{I}}$

In this section we discuss the equivalence between the LTS $\mathcal{D}_{\mathcal{I}}$, presented in Section 8, and the LTS $\mathcal{S}_{\mathcal{I}}$, introduced in Section 9. In particular, we provide a proof of Theorem 9.1, for which we introduce the following two useful propositions.

Proposition A.1. Let P be a pure process. If

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{D}} Q_\epsilon,$$

then there exists a linear process Q'_ϵ such that

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q'_\epsilon$$

and for each substitution σ, we have $Q_\epsilon\sigma \equiv Q'_\epsilon\sigma$.

Proof (sketch). The proof is by cases on the rules giving

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{D}} Q_\epsilon.$$

We begin by observing that the rules in Figure 22 and the rules in the first two rows of Figure 24 derive exactly the same transition relation for the reduction relation of mobile ambients, so the proposition holds trivially for them.

For the rules in Figure 23, we will just show the proof for the In rule as an example. So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{D}} Q_\epsilon$$

by the In rule. This means that

$$\begin{aligned} P &\equiv (\nu A)(in\ m.P_1|P_2), \qquad m \notin A \\ Q_\epsilon &= (\nu A)(m[x[P_1|P_2|X_1]|X_2]) \\ C_\epsilon[-] &= x[-|X_1]|m[X_2]. \end{aligned}$$

We have to show that there exists a linear process Q'_ϵ such that

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q'_\epsilon$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

and for each substitution σ, we have

$$Q_\epsilon \sigma \equiv Q'_\epsilon \sigma.$$

In order to show this, we proceed by cases on the rules giving

$$P \equiv (\nu A)(in\ m.P_1 | P_2).$$

As an example, we will just consider the rule $P \equiv P$.

Note that by applying the 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 the InPar rule to get

$$in\ m.P_1|P_2 \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{S}} m[x[P_1|P_2|X_1]|X_2].$$

Since we also know $m \notin A$, from the InRes rule, we can conclude

$$(\nu A)(in\ m.P_1|P_2) \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{S}} Q_\epsilon,$$

so

$$P \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{S}} Q_\epsilon,$$

and thus, trivially, for each substitution σ, we have $Q_\epsilon \sigma \equiv Q_\epsilon \sigma$. □

Proposition A.2. Let P be a pure process. If

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q_\epsilon,$$

then there exists a linear process Q'_ϵ such that

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{D}} Q'_\epsilon$$

and for each substitution σ, we have $Q_\epsilon \sigma \equiv Q'_\epsilon \sigma$.

Proof (sketch). The proof is by cases on the rules giving

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q_\epsilon.$$

As in the proof of the previous proposition, the proposition holds trivially for the rules in the first two rows of Figure 24. For the remaining rules in Figure 24, we will just show the cases for the In and InPar rules as an example:

— In rule:

We assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by the In rule. This means that

$$P = in\ m.P_1$$
$$Q_\epsilon = m[x[P_1|X_1]|X_2]$$
$$C_\epsilon[-] = x[-|X_1]|m[X_2].$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

It is easy to check that

$$P \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{D}} Q_\epsilon$$

by the IN rule, so the proposition trivially holds.

— INPAR rule:
We assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by the INPAR rule. This means that

$$P = P'|R'$$
$$C_\epsilon[-] = x[-|X_1]|m[X_2]$$
$$Q_\epsilon = Q''_\epsilon \left\{ {}^{R'|X_1}/_{X_1} \right\}$$

and

$$P' \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q'_\epsilon.$$

By the induction hypothesis, we have

$$P' \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{D}} Q''_\epsilon,$$

which means that

$$P' \equiv (\nu A)(in\ m.P_1|P_2)$$
$$Q''_\epsilon = (\nu A)(m[x[P_1|P_2|X_1]|X_2]).$$

with $m \notin A$.
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 that

$$(\nu A)(in\ m.P_1|P_2)$$

is α-equivalent to

$$(\nu A')(in\ m.P'_1|P'_2)$$

and

$$A' \cap fn(R') = \varnothing.$$

So, using the IN rule,

$$P'|R' \xrightarrow{x[-|X_1]|m[X_2]}_{\mathcal{D}} Q'_\epsilon,$$

where

$$Q'_\epsilon = (\nu A')(m[x[P'_1|P'_2|R'|X_1]|X_2]).$$

It is then easy to check that for each substitution σ, we have $Q_\epsilon\sigma \equiv Q'_\epsilon\sigma$. □

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Theorem 9.1 follows trivially from these two propositions and from Definitions 8.1 and 9.1.

Appendix B. Correspondence between the LTS $\mathcal{S}_{\mathcal{I}}$ and the LTS $\mathcal{CA}$

In this section we prove Propositions 10.1 and 10.2, which are used to give a formal proof of the correspondence between our LTS $\mathcal{S}_I$ defined on pure processes of mobile ambients and the LTS $\mathcal{CA}$ for mobile ambients proposed in Rathke and Sobociński (2008b).

B.1. *Proof of Proposition 10.1*

We will begin with the proof of Proposition 10.1, which is needed to prove the first statement of Theorem 10.1.

Proof of Proposition 10.1. The proof is by cases on the rules giving

$$P \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}}_{\mathcal{CA}} Q.$$

— $C\lambda$ rule:
We assume that

$$P \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}}_{\mathcal{CA}} Q$$

by the $C\lambda$ rule. This means that

$$P \xrightarrow{\alpha}_{\mathcal{C}} A$$
$$A \xrightarrow{\vec{M}^{\alpha}\downarrow}_{\mathcal{A}} Q$$

with $\alpha \notin \{[\overline{in\, m}], \overline{open\, n}, \tau\}$.
We now proceed by cases on the rules giving

$$P \xrightarrow{\alpha}_{\mathcal{C}} A$$

with $\alpha \notin \{[\overline{in\, m}], \overline{open\, n}, \tau\}$:

– In, InAmb, Ou and Op:
These cases are analogous, so we will only show the case of the In rule as an example.
So we assume that $P \xrightarrow{\alpha}_{\mathcal{C}} A$ by the In rule. This means that

$$P = in\ m.P_1$$
$$A = \lambda X x Y .m[x[P_1|X]|Y]$$
$$\alpha = in\ m.$$

We assume that $\vec{M}^{\alpha} = R, n, S$, for R and S processes and n an ambient name, so we have

$$Q = m[n[P_1|R]|S].$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

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

$$\sigma_M^\alpha = \left\{ {}^{R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\}.$$

We take

$$Q_\epsilon = m[x[P_1|X_1]|X_2].$$

It is easy to check that

$$in\ m.P_1 \xrightarrow{C_\epsilon^{in\,m}[-]}_{\mathcal{S}} m[x[P_1|X_1]|X_2]$$

by the In rule. Moreover, we have

$$m[x[P_1|X_1]|X_2]\sigma_M^\alpha = m[n[P_1|R]|S] \equiv Q.$$

– ||In, ||InAmb, ||Ou and ||Op:
The cases for these rules are analogous, so we will just show the case of the ||In rule as an example.
So we assume that $P \xrightarrow{\alpha}_{\mathcal{C}} A$ by the ||In rule. This means that

$$\begin{aligned} \alpha &= in\ m \\ P &= P_1|P_2 \\ A &= \lambda X.A'(P_2|X) \end{aligned}$$

and

$$P_1 \xrightarrow{in\,m}_{\mathcal{C}} A'.$$

We assume that $\vec{M}^\alpha = R, n, S$, for R and S processes and n an ambient name, so we have

$$Q = A'(P_2|R, n, S).$$

Let us consider

$$\vec{M}'^\alpha = P_2|R, n, S.$$

Since

$$A \xrightarrow{\vec{M}^\alpha\downarrow}_{\mathcal{A}} Q,$$

we have

$$A' \xrightarrow{\vec{M}'^\alpha\downarrow}_{\mathcal{A}} Q.$$

Therefore, we have

$$P_1 \xrightarrow{in\,m\downarrow\vec{M}'^\alpha}_{\mathcal{C}\mathcal{A}} Q.$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

By the induction hypothesis, there exists Q'_ϵ such that

$$P_1 \xrightarrow{C_\epsilon^{in\,m}[-]}_{\mathcal{S}} Q'_\epsilon$$

and

$$Q \equiv Q'_\epsilon \sigma^\alpha_{M'},$$

where

$$\sigma^\alpha_{M'} = \left\{ {}^{P_2|R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\}.$$

We have to show that there exists Q_ϵ, such that

$$P \xrightarrow{C_\epsilon^{in\,m}[-]}_{\mathcal{S}} Q_\epsilon$$

and

$$Q \equiv Q_\epsilon \sigma^\alpha_M,$$

with

$$\sigma^\alpha_M = \left\{ {}^{R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\}.$$

We take

$$Q_\epsilon = Q'_\epsilon \left\{ {}^{P_2|X_1}/_{X_1} \right\}.$$

It is easy to check that

$$P_1|P_2 \xrightarrow{C_\epsilon^{in\,m}[-]}_{\mathcal{S}} Q'_\epsilon \left\{ {}^{P_2|X_1}/_{X_1} \right\}$$

by the InPar rule. Moreover, it is obvious that

$$Q'_\epsilon \left\{ {}^{P_2|X_1}/_{X_1} \right\} \left\{ {}^{R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\} = Q'_\epsilon \left\{ {}^{P_2|R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\} \equiv Q.$$

– *v*In, *v*InAmb, *v*Ou and *v*Op:
These cases are analogous, so we will just show the case of the *v*In rule as an example.
So we assume that

$$P \xrightarrow{\alpha}_{\mathcal{C}} A$$

by the *v*In rule. This means that

$$\begin{aligned} \alpha &= in\ m \\ P &= (\nu n) P_1 \\ A &= (\nu n) A' \end{aligned}$$

and

$$P_1 \xrightarrow{in\,m}_{\mathcal{C}} A' \qquad \text{with } m \neq n.$$

We assume that $\vec{M}^\alpha = R, n, S$, for R and S processes and n an ambient name, so we have

$$Q = A(R, n, S).$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Since we know that

$$P_1 \xrightarrow{in\, m}_{\mathcal{C}} A'$$

and

$$A' \xrightarrow{\vec{M}^{\alpha}\downarrow}_{\mathcal{A}} A'(R, n, S) = Q',$$

we have

$$P_1 \xrightarrow{in\, m \downarrow \vec{M}^{\alpha}}_{\mathcal{CA}} Q'.$$

So, by the 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

$$\sigma^{\alpha}_{M} = \left\{ {}^{R}/_{X_1}, {}^{n}/_{x}, {}^{S}/_{X_2} \right\}.$$

We now have to show that there exists Q_{ϵ} such that

$$P \xrightarrow{C^{in\, m}_{\epsilon}[-]}_{\mathcal{S}} Q_{\epsilon}$$

and

$$Q \equiv Q_{\epsilon}\sigma^{\alpha}_{M}.$$

We take $Q_{\epsilon} = (\nu n)Q'_{\epsilon}$. It is then easy to check that, since $n \neq m$, we have

$$(\nu n)P_1 \xrightarrow{C^{in\, m}_{\epsilon}[-]}_{\mathcal{S}} (\nu n)Q'_{\epsilon}$$

by the InRes rule and

$$Q \equiv Q\epsilon\sigma^{\alpha}_{M}.$$

This completes the proof for the $C\lambda$ rule.

— coInλ and coOpλ:
These cases are analogous, so we will just show the case of the coInλ rule as an example.
So we assume that

$$P \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}}_{\mathcal{CA}} Q$$

by the coInλ rule. This means that

$$\alpha = [\overline{in\, m}]$$
$$\vec{M}^{\alpha} = R, S, n$$

and

$$P \xrightarrow{[\overline{in\, m}]}_{\mathcal{C}} A$$
$$A(\lambda X\, Y\, Z\, x.m[x[Y\,|Z]|X]) \xrightarrow{R,S,n\downarrow}_{\mathcal{A}} Q.$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

We now proceed by cases on the rules giving

$$P \xrightarrow{\overline{[in\, m]}}_{\mathcal{C}} A.$$

– coIn:
We assume that

$$P \xrightarrow{\overline{[in\, m]}}_{\mathcal{C}} A$$

by the coIn rule. This means that $P = m[P_1]$ and $A = \lambda Z.Z(P_1)$, so we have

$$Q = m[n[R|S]|P_1].$$

We have to show that there exists Q_ϵ, such that

$$P \xrightarrow{C_\epsilon^{\overline{[in\, m]}}[-]}_{\mathcal{S}} Q_\epsilon$$

and

$$Q \equiv Q_\epsilon \sigma_M^\alpha,$$

with

$$\sigma_M^\alpha = \left\{ {}^{R}/_{X_1}, {}^{S}/_{X_2}, {}^{n}/_{x} \right\}.$$

We take

$$Q_\epsilon = m[x[X_1|X_2]|P_1].$$

It is then easy to check that

$$m[P_1] \xrightarrow{C_\epsilon^{\overline{[in\, m]}}[-]}_{\mathcal{S}} m[x[X_1|X_2]|P_1]$$

by the CoIn rule and that

$$m[x[X_1|X_2]|P_1] \left\{ {}^{R}/_{X_1}, {}^{S}/_{X_2}, {}^{n}/_{x} \right\} = m[n[R|S]|P_1] \equiv Q.$$

– ||coIn:
So we assume that

$$P \xrightarrow{\overline{[in\, m]}}_{\mathcal{C}} A$$

by the ||coIn rule. This means that

$$P = P_1|P_2$$
$$P_1 \xrightarrow{\overline{[in\, m]}}_{\mathcal{C}} A'$$
$$A = A'|P_2.$$

Since

$$P_1 \xrightarrow{\overline{[in\, m]}}_{\mathcal{C}} A'$$

and

$$A'(\lambda X Y Z x.m[x[Y|Z]|X]) \xrightarrow{R,S,n\downarrow}_{\mathcal{A}} Q'$$

such that

$$Q = Q'|P_2,$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

we have that

$$P_1 \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}}_{\mathcal{CA}} Q'.$$

So, by the induction hypothesis,

$$P_1 \xrightarrow{C_{\epsilon}^{\overline{[in\,m]}}[-]}_{\mathcal{S}} Q'_{\epsilon}$$

with

$$Q' \equiv Q'_{\epsilon}\sigma_M^{\alpha}$$

with

$$\sigma_M^{\alpha} = \left\{ {}^{R}/_{X_1}, {}^{S}/_{X_2}, {}^{n}/_{x} \right\}.$$

We have to show that there exists Q_{ϵ}, such that

$$P \xrightarrow{C_{\epsilon}^{\overline{[in\,m]}}[-]}_{\mathcal{S}} Q_{\epsilon}$$

and

$$Q \equiv Q_{\epsilon}\sigma_M^{\alpha}.$$

We take $Q_{\epsilon} = Q'_{\epsilon}|P_2$. It is then easy to check that

$$P \xrightarrow{C_{\epsilon}^{\overline{[in\,m]}}[-]}_{\mathcal{S}} Q_{\epsilon}$$

by the CoInPar rule. Moreover,

$$\begin{aligned} Q &= Q'|P_2 \\ &\equiv Q'_{\epsilon}\sigma_M^{\alpha}|P_2 \\ &= (Q'_{\epsilon}|P_2)\sigma_M^{\alpha} \\ &= Q_{\epsilon}\sigma_M^{\alpha}. \end{aligned}$$

– *v*CoIn rule:
So we assume that

$$P \xrightarrow{\overline{[in\,m]}}_{\mathcal{C}} A$$

by the *v*CoIn rule. This means that

$$\begin{aligned} P &= (vn)P_1 \\ A &= (v)A' \\ n &\neq m \end{aligned}$$

and

$$P_1 \xrightarrow{\overline{[in\,m]}}_{\mathcal{C}} A'.$$

Since

$$P_1 \xrightarrow{\overline{[in\,m]}}_{\mathcal{C}} A'$$

and

$$A'(\lambda XYZx.m[x[Y|Z]|X]) \xrightarrow{R,S,n\downarrow}_{\mathcal{A}} Q'$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

such that $Q = (\nu)Q'$, we have that

$$P_1 \xrightarrow{\alpha\downarrow\vec{M}^{\alpha}}_{\mathcal{CA}} Q'.$$

So, by the induction hypothesis,

$$P_1 \xrightarrow{C_{\epsilon}^{[\overline{in}\, m]}[-]}_{\mathcal{S}} Q'_{\epsilon}$$

with

$$Q' \equiv Q'_{\epsilon}\sigma_M^{\alpha}$$

with

$$\sigma_M^{\alpha} = \left\{ {}^{R}/_{X_1}, {}^{S}/_{X_2}, {}^{n}/_{x} \right\}.$$

We have to show that there exists Q_{ϵ}, such that

$$P \xrightarrow{C_{\epsilon}^{[\overline{in}\, m]}[-]}_{\mathcal{S}} Q_{\epsilon}$$

and

$$Q \equiv Q_{\epsilon}\sigma_M^{\alpha}.$$

We take $Q_{\epsilon} = (\nu n)Q'_{\epsilon}$. It is then easy to check that

$$P \xrightarrow{C_{\epsilon}^{[\overline{in}\, m]}[-]}_{\mathcal{S}} Q_{\epsilon}$$

by the CoInRes rule. Moreover,

$$\begin{aligned} Q &= (\nu n)Q' \\ &\equiv (\nu n)(Q'_{\epsilon}\sigma_M^{\alpha}) \\ &= ((\nu n)Q'_{\epsilon})\sigma_M^{\alpha} \\ &= Q_{\epsilon}\sigma_M^{\alpha}. \end{aligned}$$

This completes the proof for the coInλ rule.

— CTau rule:
So we assume that $P \xrightarrow{\alpha}_{\mathcal{CA}} Q$ by the CTau rule. This means that $P \to Q$ by Rathke and Sobociński (2008b, Proposition 7), so

$$P \xrightarrow{C_{\epsilon}^{\tau}}_{\mathcal{S}} Q,$$

with $C_{\epsilon}^{\tau} = -$. □

B.2. *Proof of Proposition 10.2*

We will now prove Proposition 10.2, which is needed to prove the second statement of Theorem 10.1.

Proof of Proposition 10.2. The proof proceeds by cases on the rules giving

$$P \xrightarrow{C_{\epsilon}[-]}_{\mathcal{S}} Q_{\epsilon}.$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

— In, InAmb, Out, OutAmb and Open:
These cases are analogous, so we will just show the case for In as an example.
So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by the In rule. This means that

$$\begin{aligned} \alpha &= in\ m \\ P &= in\ m.P_1 \\ C_\epsilon^{in\ m}[-] &= x[-|X_1]|m[X_2] \\ Q_\epsilon &= m[x[P_1|X_1]|X_2]. \end{aligned}$$

Moreover, the substitution σ has the shape

$$\left\{ {}^{P_2}/_{X_1}, {}^{n}/_{x}, {}^{P_3}/_{X_2} \right\}$$

for some ambient name n and some processes P_1 and P_2. Therefore, we have

$$\begin{aligned} Q &\equiv Q_\epsilon \sigma \\ &= m[x[P_1|X_1]|X_2] \left\{ {}^{P_2}/_{X_1}, {}^{n}/_{x}, {}^{P_3}/_{X_2} \right\} \\ &= m[n[P_1|P_2]|P_3]. \end{aligned}$$

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

$$P \xrightarrow{in\ m}_{\mathcal{C}} \lambda X x Y .m[x[P_1|X]|Y]$$

because of the In rule in Rathke and Sobociński (2008b, Figure 6). Moreover, we can apply the Inst rule in Rathke and Sobociński (2008b, Figure 7), 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, from the Cλ rule in Rathke and Sobociński (2008b, Figure 8), we can conclude

$$P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} m[n[P_1|P_2]|P_3],$$

and

$$P \xrightarrow{in\ m \downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q$$

by the StrCng rule.

— InPar, InAmbPar, OutPar, OutAmbPar and OpenPar:
These cases are analogous, so we will just show the case for InPar as an example.
So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

by the InPar rule. This means that

$$\alpha = in\ m$$
$$P = P_1|Q_1$$
$$C_\epsilon^{in\ m}[-] = x[-|X_1]|m[X_2]$$
$$Q_\epsilon = P_\epsilon \left\{ {}^{Q_1|X_1}/_{X_1} \right\}$$

and

$$P_1 \xrightarrow{C_\epsilon^{in\ m}[-]}_{\mathcal{S}} P_\epsilon.$$

Moreover, the substitution σ has the shape

$$\left\{ {}^{P_2}/_{X_1}, {}^{n}/_{x}, {}^{P_3}/_{X_2} \right\}$$

for some ambient name n and some processes P_1 and P_2. Consider the substitution

$$\sigma' = \left\{ {}^{Q_1|P_2}/_{X_1}, {}^{n}/_{x}, {}^{P_3}/_{X_2} \right\}.$$

Note that

$$P_\epsilon\sigma' = Q_\epsilon\sigma \equiv Q.$$

Since

$$P_1 \xrightarrow{C_\epsilon^{in\ m}[-]}_{\mathcal{S}} P_\epsilon,$$

by applying the induction hypothesis, we have

$$P_1 \xrightarrow{in\ m \downarrow \vec{M}^\alpha_{\sigma'}}_{\mathcal{CA}} P_\epsilon\sigma',$$

where

$$\vec{M}^\alpha_{\sigma'} = Q_1|P_2, n, P_3.$$

We have to show that

$$P \xrightarrow{in\ m \downarrow \vec{M}^\alpha_{\sigma}}_{\mathcal{CA}} Q,$$

where $\vec{M}^\alpha_\sigma = P_2, n, P_3$. We know that

$$P_1 \xrightarrow{in\ m \downarrow \vec{M}^\alpha_{\sigma'}}_{\mathcal{CA}} P_\epsilon\sigma'.$$

This means that

$$P_1 \xrightarrow{in\ m}_{\mathcal{C}} A$$

and

$$A \xrightarrow{\vec{M}^\alpha_{\sigma'} \downarrow}_{\mathcal{A}} P_\epsilon\sigma'.$$

Since

$$P_1 \xrightarrow{in\ m}_{\mathcal{C}} A,$$

because of the ||In rule in Rathke and Sobociński (2008b, Figure 6), we have

$$P_1|Q_1 \xrightarrow{in\ m}_{\mathcal{C}} \lambda X.A(Q_1|X).$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

It is then easy to check that if

$$A \xrightarrow{\vec{M}^{\alpha}_{\sigma'}\downarrow}_{\mathcal{A}} P_{\epsilon}\sigma',$$

then we also have

$$\lambda X.A(Q_1|X) \xrightarrow{\vec{M}^{\alpha}_{\sigma}\downarrow}_{\mathcal{A}} P_{\epsilon}\sigma'.$$

Therefore, by applying the Cλ rule in Rathke and Sobociński (2008b, Figure 8), we get

$$P_1|Q_1 \xrightarrow{in\, m\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} P_{\epsilon}\sigma'$$

and

$$P \xrightarrow{in\, m\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} Q$$

by the StrCng rule.

— InRes, InAmbRes, OutRes, OutAmbRes and OpenRes:
These cases are analogous, so we will just show the case for InRes as an example. So we assume that

$$P \xrightarrow{C_{\epsilon}[-]}_{\mathcal{S}} Q_{\epsilon}$$

by the InRes rule. This means that

$$\begin{aligned} \alpha &= in\, m \\ P &= (va)P_1 \\ a &\neq m \\ C^{in\, m}_{\epsilon}[-] &= x[-|X_1]|m[X_2] \\ Q_{\epsilon} &= (va)P_{\epsilon} \end{aligned}$$

and

$$P_1 \xrightarrow{C^{in\, m}_{\epsilon}[-]}_{\mathcal{S}} P_{\epsilon}.$$

Moreover, the substitution σ, avoiding name capture, has the shape

$$\left\{{}^{P_2}/_{X_1}, {}^{n}/_{x}, {}^{P_3}/_{X_2}\right\}$$

for some ambient name n and some processes P_1 and P_2. Therefore,

$$(va)P_{\epsilon}\sigma = Q_{\epsilon}\sigma \equiv Q.$$

Since

$$P_1 \xrightarrow{C^{in\, m}_{\epsilon}[-]}_{\mathcal{S}} P_{\epsilon},$$

by applying the induction hypothesis, we have

$$P_1 \xrightarrow{in\, m\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} P_{\epsilon}\sigma,$$

where $\vec{M}^{\alpha}_{\sigma} = P_2, n, P_3$. We have to show that

$$P \xrightarrow{in\, m\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} Q.$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

We know that

$$P_1 \xrightarrow{in\, m \downarrow \vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} P_\epsilon \sigma,$$

so

$$P_1 \xrightarrow{in\, m}_{\mathcal{C}} A$$

and

$$A \xrightarrow{\vec{M}^{\alpha}_{\sigma} \downarrow}_{\mathcal{A}} P_\epsilon \sigma.$$

Since

$$P_1 \xrightarrow{in\, m}_{\mathcal{C}} A,$$

because of the *v*In rule in Rathke and Sobociński (2008b, Figure 6), we have

$$(va)P_1 \xrightarrow{in\, m}_{\mathcal{C}} (va)A$$

since $a \neq m$. It is easy to check that since

$$A \xrightarrow{\vec{M}^{\alpha}_{\sigma} \downarrow}_{\mathcal{A}} P_\epsilon \sigma,$$

we have

$$(va)A \xrightarrow{\vec{M}^{\alpha}_{\sigma} \downarrow}_{\mathcal{A}} (va)P_\epsilon \sigma.$$

Therefore, applying the Cλ rule in Rathke and Sobociński (2008b, Figure 8), we get

$$(va)P_1 \xrightarrow{in\, m \downarrow \vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} (va)P_\epsilon \sigma,$$

and

$$P \xrightarrow{in\, m \downarrow \vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} Q$$

by the StrCng rule.

— CoIn and CoOpen:
These cases are analogous, so we will just show the case for CoIn as an example. So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by CoIn rule. This means that

$$\begin{aligned} \alpha &= [\overline{in\ m}] \\ P &= m[P_1] \\ C_\epsilon^{[\overline{in\ m}]}[-] &= -|x[in\ m.X_1|X_2] \\ Q_\epsilon &= m[x[X_1|X_2]|P_1]. \end{aligned}$$

Moreover, the substitution σ has the shape

$$\left\{ {}^{P_2}/_{X_1}, {}^{P_3}/_{X_2}, {}^{n}/_{x} \right\}$$

for some ambient name n and some processes P_1 and P_2. So we have

$$Q \equiv Q_\epsilon \sigma = m[n[P_2|P_3]|P_1].$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

We have to show that

$$P \xrightarrow{[\overline{in\ m}]\downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q,$$

where $\vec{M}_\sigma^\alpha = P_2, P_3, n$. It is easy to check that

$$P \xrightarrow{[\overline{in\ m}]}_{\mathcal{C}} \lambda Z.Z(P_1),$$

by the CoIn rule in Rathke and Sobociński (2008b, Figure 6). Moreover, by the Inst rule in Rathke and Sobociński (2008b, Figure 7), we have

$$(\lambda Z.Z(P_1))(\lambda XYZx.m[x[Y|Z]|X]) \xrightarrow{\vec{M}_\sigma^\alpha\downarrow}_{\mathcal{A}} m[n[P_2|P_3]|P_1].$$

Therefore, because of the CoInλ rule in Rathke and Sobociński (2008b, Figure 8), we can conclude

$$P \xrightarrow{[\overline{in\ m}]\downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} m[n[P_2|P_3]|P_1],$$

and

$$P \xrightarrow{[\overline{in\ m}]\downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q$$

by the StrCng rule.

— CoInPar and CoOpenPar:
These cases are analogous, so we will just show the case for CoInPar as an example. So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by the CoInPar rule. This means that

$$\alpha = [\overline{in\ m}]$$
$$P = P_1|Q_1$$
$$C_\epsilon^{[\overline{in\ m}]}[-] = -|x[in\ m.X_1|X_2]$$
$$Q_\epsilon = P_\epsilon|Q_1.$$

and

$$P_1 \xrightarrow{C_\epsilon^{[\overline{in\ m}]}[-]}_{\mathcal{S}} P_\epsilon.$$

Moreover, the substitution σ has the shape

$$\left\{ {}^{P_2}/_{X_1}, {}^{P_3}/_{X_2}, {}^{n}/_{x} \right\}$$

for some ambient name n and some processes P_1 and P_2. So we have

$$Q \equiv Q_\epsilon\sigma = P_\epsilon|Q_1\sigma = P_\epsilon\sigma|Q_1.$$

We have to show that

$$P \xrightarrow{[\overline{in\ m}]\downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} Q,$$

where $\vec{M}_\sigma^\alpha = P_2, P_3, n$. By the induction hypothesis,

$$P_1 \xrightarrow{[\overline{in\ m}]\downarrow \vec{M}_\sigma^\alpha}_{\mathcal{CA}} P_\epsilon\sigma$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

by the CoInλ rule in Rathke and Sobociński (2008b, Figure 8). This means that

$$P_1 \xrightarrow{[\overline{in\ m}]}_{\mathcal{C}} A$$

and

$$A(\lambda X Y Z x.n[x[Y|Z]|X]) \xrightarrow{\vec{M}^{z}_{\sigma}\downarrow}_{\mathcal{A}} P_\epsilon \sigma.$$

Therefore, by the ||CoIn rule of Rathke and Sobociński (2008b, Figure 6), we have

$$P_1|Q_1 \xrightarrow{[\overline{in\ m}]}_{\mathcal{C}} A|Q_1$$

and

$$A|Q_1(\lambda X Y Z x.n[x[Y|Z]|X]) \xrightarrow{\vec{M}^{z}_{\sigma}\downarrow}_{\mathcal{A}} P_\epsilon \sigma|Q_1.$$

So we can conclude that

$$P_1|Q_1 \xrightarrow{[\overline{in\ m}]\downarrow\vec{M}^{z}_{\sigma}}_{\mathcal{CA}} P_\epsilon \sigma|Q_1$$

by the CoInλ rule, and

$$P \xrightarrow{[\overline{in\ m}]\downarrow\vec{M}^{z}_{\sigma}}_{\mathcal{CA}} Q$$

by the StrCng rule.

— CoInRes and CoOpenRes:
These cases are analogous, so we will just show the case for CoInRes as an example. So we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by the CoInRes rule. This means that

$$\alpha = [\overline{in\ m}]$$
$$P = (\nu a)P_1$$
$$a \neq n$$
$$C_\epsilon^{[\overline{in\ m}]}[-] = -|x[in\ m.X_1|X_2]$$
$$Q_\epsilon = (\nu a)P_\epsilon.$$

and

$$P_1 \xrightarrow{C_\epsilon^{[\overline{in\ m}]}[-]}_{\mathcal{S}} P_\epsilon$$

Moreover, the substitution σ, avoiding name capture, has the shape

$$\left\{{}^{P_2}/_{X_1}, {}^{P_3}/_{X_2}, {}^{n}/_{x}\right\}$$

for some ambient name n and some processes P_1 and P_2. So, we have

$$Q \equiv Q_\epsilon \sigma = ((\nu a)P_\epsilon)\sigma = (\nu a)P_\epsilon \sigma.$$

We have to show that

$$P \xrightarrow{[\overline{in\ m}]\downarrow\vec{M}^{z}_{\sigma}}_{\mathcal{CA}} Q,$$

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

where $\vec{M}^{\alpha}_{\sigma} = P_2, P_3, n$. By the induction hypothesis,

$$P_1 \xrightarrow{[\overline{in\, m}]\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} P_\epsilon\sigma$$

by the CoInλ rule in Rathke and Sobociński (2008b, Figure 8). This means that

$$P_1 \xrightarrow{[\overline{in\, m}]}_{\mathcal{C}} A$$

and

$$A(\lambda X Y Z x.n[x[Y\,|Z]|X]) \xrightarrow{\vec{M}^{\alpha}_{\sigma}\downarrow}_{\mathcal{A}} P_\epsilon\sigma.$$

So, by the νCoIn rule of Rathke and Sobociński (2008b, Figure 6), we have

$$(\nu a)P_1 \xrightarrow{[\overline{in\, m}]}_{\mathcal{C}} (\nu a)A$$

and

$$(\nu a)A(\lambda X Y Z x.n[x[Y\,|Z]|X]) \xrightarrow{\vec{M}^{\alpha}_{\sigma}\downarrow}_{\mathcal{A}} (\nu a)P_\epsilon\sigma.$$

So we can conclude that

$$P \xrightarrow{[\overline{in\, m}]\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} (\nu a)P_\epsilon\sigma$$

by the CoInλ rule, and

$$P \xrightarrow{[\overline{in\ m}]\downarrow\vec{M}^{\alpha}_{\sigma}}_{\mathcal{CA}} Q$$

by the StrCng rule.

— InTau, OutTau, OpenTau, TauAmb, TauPar and TauRes:
These cases are analogous, so we assume that

$$P \xrightarrow{C_\epsilon[-]}_{\mathcal{S}} Q_\epsilon$$

by one of these rules. This means that $P \to Q$ and $Q_\epsilon \equiv Q$. So, from Rathke and Sobociński (2008b, Proposition 7), we have

$$P \xrightarrow{\tau}_{\mathcal{CA}} Q,$$

which completes the proof. □

Appendix C. A standard SOS presentation of LTS $\mathcal{S}$

In this section we present a standard SOS specification equivalent to the one for the LTS $\mathcal{S}$. The rules generating the LTS $\mathcal{N}$ are obtained from those for $\mathcal{S}$ (see Figure 24) by adding the rules in Figure 25 (where rules with the same name are replacements), and requiring that

$$\to_I \cup \to_O \subseteq \to_{\mathcal{N}}.$$

The main difference between the LTSs $\mathcal{S}$ and $\mathcal{N}$ is that in the former the reduction relation is given by using the reduction rules labelled with the identity context $-$, while in

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

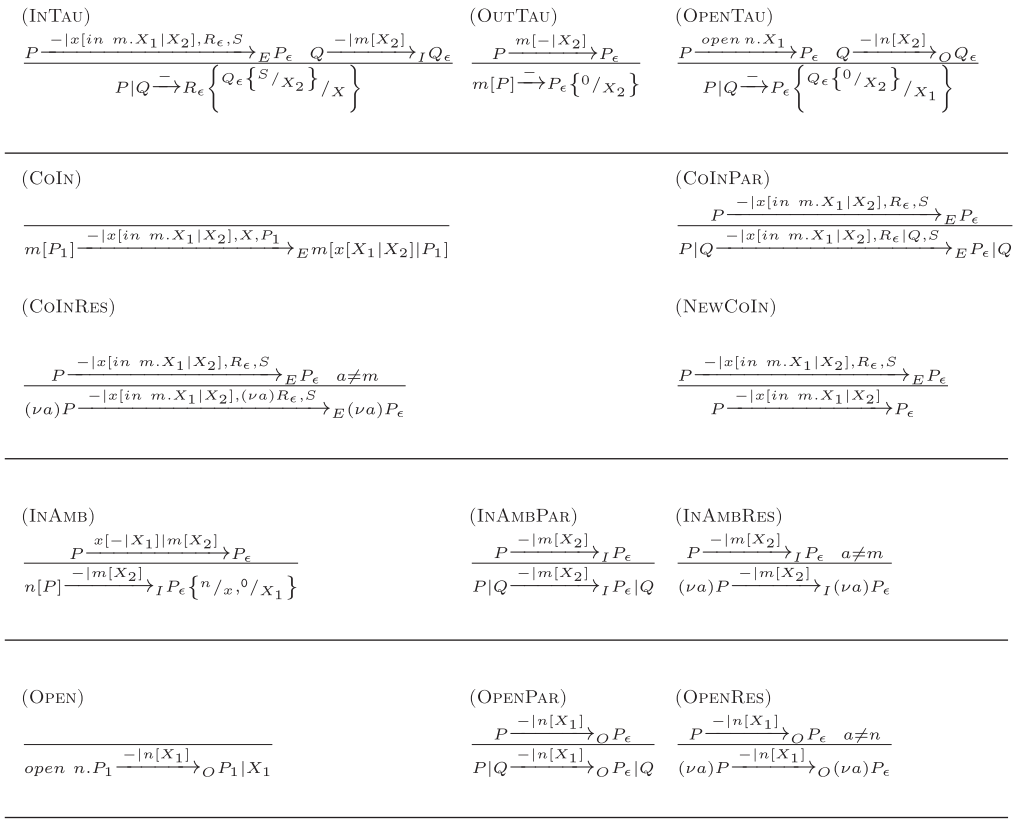


Fig. 25. The LTS $\mathcal{N}$.

the latter the reduction relation is defined inductively. However, to do this for the InTau rule, it is not enough to use the synchronisation between the transitions

$$P \xrightarrow{-|x[in\ m.X_1|X_2]} P_\epsilon$$

and

$$Q \xrightarrow{-|m[X_2]} Q_\epsilon,$$

which must be derived by applying the rules in the fourth row of Figure 24. This is for two reasons:

(1) the label $-|m[X_2]$ can be also obtained by applying the rules in the eighth row;
(2) the resulting state of the InTau rule cannot be obtained using a trivial combination of P_ϵ and Q_ϵ.

Both problems can be solved by moving beyond the 'context as labels' philosophy we have adopted for the other LTSs in the paper. More precisely, the first problem can be easily solved by recording which rule has been applied. However, to address the second,

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

we have to construct an open process that allows us to obtain the resulting state by simply replacing a suitable instance of Q_ϵ in it.

The following proposition states the relation between the LTSs $\mathcal{S}$ and $\mathcal{N}$.

Proposition C.1. Let P be a pure process. If

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{N}} Q_\epsilon,$$

then there exists a linear process Q'_ϵ such that

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q'_\epsilon$$

and for each substitution σ, we have

$$Q_\epsilon\sigma \equiv Q'_\epsilon\sigma.$$

Conversely, if

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{S}} Q_\epsilon,$$

then there exists a linear process Q'_ϵ such that

$$P \xrightarrow{C[-]_\epsilon}_{\mathcal{N}} Q'_\epsilon$$

and for each substitution σ, we have

$$Q_\epsilon\sigma \equiv Q'_\epsilon\sigma.$$

References

Baldan, P., Corradini, A., Ehrig, H., Löwe, M., Montanari, U. and Rossi, F. (1999) Concurrent semantics of algebraic graph transformation. In: Ehrig, H., Kreowski, H.-J., Montanari, U. and Rozenberg, G. (eds.) Concurrency, Parallelism, and Distribution. *Handbook of Graph Grammars and Computing by Graph Transformation 3: Concurrency, Parallelism, and Distribution*, World Scientific 107–187.

Baldan, P., Ehrig, H. and König, B. (2006) Composition and decomposition of DPO transformations with borrowed context. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L. and Rozemberg, G. (eds.) Graph Transformation. *Springer-Verlag Lecture Notes in Computer Science* **4178** 153–167.

Bonchi, F. (2008) *Abstract Semantics by Observable Contexts*, Ph.D. thesis, Department of Informatics, University of Pisa.

Bonchi, F. and Montanari, U. (2007) Coalgebraic models for reactive systems. In: Caires, L. and Vasconcelos, V. (eds.) Concurrency Theory. *Springer-Verlag Lecture Notes in Computer Science* **4703** 364–379.

Bonchi, F., Gadducci, F. and König, B. (2006a) Process bisimulation via a graphical encoding. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L. and Rozemberg, G. (eds.) Graph Transformation. *Springer-Verlag Lecture Notes in Computer Science* **4178** 168–183.

Bonchi, F., Gadducci, F. and Monreale, G. V. (2009a) Labelled transitions for mobile ambients (as synthesized via a graphical encoding). In: Hildebrandt, T. and Gorla, D. (eds.) Expressiveness in Concurrency. *Electronic Notes in Theoretical Computer Science* **242** (1) 73–98.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Bonchi, F., Gadducci, F. and Monreale, G. V. (2009b) On barbs and labels in reactive systems. In: Klin, B. and Sobociński, P. (eds.) Structural Operational Semantics. *Electronic Proceedings in Theoretical Computer Science* **18** 46–61.

Bonchi, F., Gadducci, F. and Monreale, G. V. (2009c) Reactive systems, barbed semantics, and the mobile ambients. In: de Alfaro, L. (ed.) Foundations of Software Science and Computation Structures. *Springer-Verlag Lecture Notes in Computer Science* **5504** 272–287.

Bonchi, F., König, B. and Montanari, U. (2006b) Saturated semantics for reactive systems. In: *Logic in Computer Science*, IEEE Computer Society 69–80.

Cardelli, L. and Gordon, A. (2000) Mobile ambients. *Theoretical Computer Science* **240** (1) 177–213.

Corradini, A. and Gadducci, F. (1999) An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures* **7** (4) 299–331.

Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. (2006) *Fundamentals of Algebraic Graph Transformation*, Springer-Verlag.

Ehrig, H. and König, B. (2006) Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. *Mathematical Structurer in Computer Science* **16** (6) 1133–1163.

Ferrari, G., Montanari, U. and Tuosto, E. (2001) A LTS semantics of ambients via graph synchronization with mobility. In: Restivo, A., Ronchi Della Rocca, S. and Roversi, L. (eds.) Italian Conference on Theoretical Computer Science. *Springer-Verlag Lecture Notes in Computer Science* **2202** 1–16.

Gadducci, F. and Monreale, G. V. (2008) A decentralized implementation of mobile ambients. In: Heckel, R. and Taentzer, G. (eds.) Graph Transformation. *Springer-Verlag Lecture Notes in Computer Science* **5214** 115–130.

Gadducci, F. and Montanari, U. (2005) Observing reductions in nominal calculi via a graphical encoding of processes. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F. and de Vrijer, R. (eds.) Processes, terms and cycles (Klop Festschrift). *Springer-Verlag Lecture Notes in Computer Science* **3838** 106–126.

Gordon, A. and Cardelli, L. (2003) Equational properties of mobile ambients. *Mathematical Structures in Computer Science* **13** (3) 371–408.

Lack, S. and Sobociński, P. (2005) Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications* **39** (3) 511–545.

Leifer, J. and Milner, R. (2000) Deriving bisimulation congruences for reactive systems. In: Palamidessi, C. (ed.) Concurrency Theory. *Springer-Verlag Lecture Notes in Computer Science* **1877** 243–258.

Merro, M. and Zappa Nardelli, F. (2005) Behavioral theory for mobile ambients. *Journal of ACM* **52** (6) 961–1023.

Milner, R. (1989) *Communication and Concurrency*, Prentice Hall.

Milner, R. (1999) *Communicating and Mobile Systems: the π-Calculus*, Cambridge University Press.

Milner, R. (2006) Pure bigraphs: Structure and dynamics. *Information and Computation* **204** (1) 60–122.

Rathke, J. and Sobociński, P. (2008a) Deconstructing behavioural theories of mobility. In: Ausiello, G., Karhumäki, J., Mauri, G. and Ong, L. (eds.) IFIP International Conference on Theoretical Computer Science. *IFIP Advances in Information and Communication Technology* **273** 507–520.

Rathke, J. and Sobociński, P. (2008b) Deriving structural labelled transitions for mobile ambients. In: van Breugel, F. and Chechik, M. (eds.) Concurrency Theory. *Springer-Verlag Lecture Notes in Computer Science* **5201** 462–476.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press

Rathke, J. and Sobocinski, P. (2010) Deriving structural labelled transitions for mobile ambients. *Information and Computation* **208** (10) 1221–1242.

Sassone, V. and Sobociṅski, P. (2005) Reactive systems over cospans. In: *Logic in Computer Science*, IEEE Computer Society 311–320.

Sobociński, P. (2004) *Deriving bisimulation congruences from reduction systems*, Ph.D. thesis, BRICS, Department of Computer Science, University of Aarhus.

https://doi.org/10.1017/S0960129512000333 Published online by Cambridge University Press