Università di Pisa
Dipartimento di Informatica

Technical Report: TR-02-14

π-Calculus Early Observational Equivalence: A First Order Coalgebraic Model

Maria Grazia Buscemi[1] Ugo Montanari[2]

August 1, 2002

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

π-Calculus Early Observational Equivalence: A First Order Coalgebraic Model

Maria Grazia Buscemi[1] and Ugo Montanari[2]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy.
[2] Dipartimento di Informatica, Università di Pisa, Italy.
buscemi@dmi.unict.it, ugo@di.unipi.it

Abstract. In this paper, we propose a compositional coalgebraic semantics of the π-calculus based on a novel approach for lifting calculi with structural axioms to coalgebraic models. We equip the transition system of the calculus with permutations, parallel composition and restriction operations, thus obtaining a bialgebra. No prefix operation is introduced, relying instead on a clause format defining the transitions of recursively defined processes. The unique morphism to the final bialgebra induces a bisimilarity relation which coincides with observational equivalence and which is a congruence with respect to the operations. The permutation algebra is enriched with a name extrusion operator δ à la De Brujin, that shifts any name to the successor and generates a new name in the first variable x_0. As a consequence, in the axioms and in the SOS rules there is no need to refer to the support, i.e., the set of significant names, and, thus, the model turns out to be first order.

1 Introduction

The π-calculus is the touchstone for several applications concerning higher order mobility, security and object orientation. The advantages of the π-calculus rely on its simplicity and its process algebra flavor. Its operational semantics is given in terms of a transition system and its abstract semantics is based on bisimilarity.

It is well known that labeled transition systems can be regarded as coalgebras for a functor in the category **Set**. A coalgebraic framework presents several advantages: morphisms between coalgebras (cohomomorphisms) enjoy the property of "reflecting behaviors" and thus they allow, for example, to characterize bisimulation equivalences as kernels of morphisms and bisimilarity as the bisimulation associated to the morphism to the final coalgebra.

However, in the above representation of transition systems, the states are seen as just forming a set, i.e., the algebraic structure modeling the construction of programs and the composition of states is disregarded.

The missing structure may be recovered by integrating coalgebras with algebras, as it is done by Turi and Plotkin [14]. They define *bialgebras* and prove

Research supported in part by FET Global project IST-2001-33100 *PROFUNDIS* and by MIUR project *COMETA*.

that for them bisimilarity is a congruence. Roughly, bialgebras are structures that can be regarded both as algebras of coalgebras and as coalgebras of algebras. Morphisms between bialgebras are both algebra homomorphisms and coalgebra cohomomorphisms and, thus, the unique morphism to the final bialgebra, which exists under reasonable assumptions, induces a (coarsest) bisimulation congruence on any coalgebra.

A fully satisfactory coalgebraic theory for the π-calculus would take advantage of well understood verification techniques and of efficient algorithms for computing finite minimal realizations [13, 12, 4].

Here our goal is to provide a compositional coalgebraic semantics of the π-calculus. We will define a permutation algebra with parallel composition and restriction for the calculus and, then, we will prove that the labeled transition system of the π-calculus can be lifted to such an algebra, thus obtaining a bialgebra.

A similar task has been considered in [9, 10]. There, a coalgebraic semantics of the π-calculus has been proposed, based on name permutations. The intuition is that the effects of permutations on the behavior of agents are the smallest information required to define an observational equivalence via ordinary bisimilarity, without restrictions due to name generation and passing. However, the proposed model is flat, i.e., it represents the calculus at an operational level, but it does not capture its algebraic structure. Rather, in the present paper we are mainly interested in a compositional interpretation of the π-calculus.

It is well known that, in the π-calculus, bisimilarity fails to be a congruence due to the input prefix. Thus, in order for this property to hold, our algebra structure will include operators of name permutation, parallel composition and restriction, but not prefix. First, we define *static* agents as π-agents whose outmost operation is either prefix, matching, recursion, or summation. We also define a bijective translation of π-agents into terms of our algebra. This function, in particular, maps static agents to constants in the algebra. Then we equip each constant with a set of reduction rules that mimic any possible transition the corresponding static agent can perform. Finally we prove that, for each π-agent, the number of static agents and associated reduction rules needed in all derivations of the agent is finite.

The compositional structure of our bialgebra also gives advantages in finite state verification methods [4]. Usually, such techniques glue components together and, then, apply model checking or other verification methods. However, in many cases, state explosion severely limits the applicability. Rather, a compositional approach gives a chance to minimize components before combining them, thus preventing state explosion, at some extent, and yielding a smaller state space.

The restriction operator of our permutation algebra has no name argument. Also extrusion and fresh input actions have no bound names. The reason is that the extruded, fresh or restricted name is assumed to be always the first one, i.e., x_0. To model name extrusion, we define a first order operator δ à la De Brujin, that shifts any name to the successor, thus generating a new name in x_0. The advantage of this choice is that it does not need a notion of support, i.e., the set of names effectively present, which would yield a second order model. Similarly, Pitts's nominal logics [11] exploits a first order model, by relying on substitution algebras that do not need considering support names.

The coalgebraic semantics for the π-calculus proposed in [9, 10] relies on the results about bialgebras in [2], where two sufficient conditions are spelled out for a labeled transition system to yield a coalgebra on the category $\mathbf{Alg}(\Gamma)$ of algebras satisfying a specification $\Gamma = (\Sigma, E)$. We follow a different strategy to prove that the transition system of the π-calculus can be lifted to yield a bialgebra. First, we consider a category $\mathbf{Alg}(\Sigma)$ of algebras where specification Σ does not contain axioms. Thus the powerset functor can be lifted to a functor on $\mathbf{Alg}(\Sigma)$ using the De Simone specification. Then, we construct a complete axiomatization (with auxiliary operators) of the π-calculus algebra and we prove that each axiom bisimulates. This is enough to ensure that the lifting can take place and thus, in particular, that bisimilarity is a congruence. The construction in [2] yields a category of bialgebras which satisfy the axioms, while in our case the algebraic specification is just a signature, without axioms. Thus our final coalgebra is larger. However, given a coalgebra which satisfies the axioms, its image in the final coalgebra is the same.

In [6], Fiore and Turi propose a semantic framework for name-passing process calculi. Their model is based on functor categories equipped with a differentiation functor δ modeling the introduction of an extra variable – the variable to be bound. Instead we work with permutation algebras enriched with the name extrusion operation δ. While we also rely on a functor category (when the algebraic specification is seen as a Lawvere theory), our setting is more uniform, since δ is at the same level of permutations and of any further operation (here parallel composition) added to the algebra. Another difference is that we deal with name permutations, rather than name substitutions, and, thus, we are able to define observational equivalence (rather than congruence) in a coalgebraic way. A difference with respect to [11] and [7] is that we assume a fixed ordering on names, while they do not need it.

The paper is organized as follows. Section 2 contains the background on permutations, π-calculus, and coalgebras. In Sect. 3 we define a category of coalgebras for mobile calculi and in Sect. 4 we introduce a permutation algebra for the π-calculus. In Sect. 5 we prove a general result to lift coalgebras in $\mathbf{Set}$ with structural axioms to coagebras on $\mathbf{Alg}(\Sigma)$ and in Sect. 6 we provide the coalgebraic semantics of the π-calculus, by applying such a result to the transition system of the calculus. Finally, Sect. 7 contains some concluding remarks.

2 Background

2.1 Names and Permutations

We need some basic definitions and properties on names and on name permutations. We denote with $\mathfrak{N} = \{x_0, x_1, x_2, \ldots\}$ the infinite, countable, totally ordered set of *names* and we use $x, y, z \ldots$ to denote names. A *name substitution* is a function $\sigma : \mathfrak{N} \to \mathfrak{N}$. We denote with $\sigma \circ \sigma'$ the composition of substitutions σ and σ'; that is, $\sigma \circ \sigma'(x) = \sigma(\sigma'(x))$. We use σ to range over substitution and we denote with $[y_1 \mapsto x_1, \cdots, y_n \mapsto x_n]$ the substitution that maps x_i into y_i for $i = 1, \ldots, n$ and which is the identity on the other names. We abbreviate by $[y \leftrightarrow x]$ the substitution $[y \mapsto x, x \mapsto y]$. The *identity substitution* is denoted by id. A *name permutation* is a bijective name substitution. We use ρ to denote a permutation. Given a permutation ρ, we define permutation ρ_{+1} as follows:

$$\frac{-}{\rho_{+1}(x_0) = x_0} \qquad\qquad \frac{\rho(x_n) = x_m}{\rho_{+1}(x_{n+1}) = x_{m+1}} \tag{1}$$

Essentially, permutation ρ_{+1} is obtained from ρ by shifting its correspondences to the right by one position.

2.2 The π-Calculus

Many versions of π-calculus have appeared in the literature. The π-calculus we present here is *early*, *monadic*, and has *synchronous* communications.

Let $\mathfrak{N}$ be the countable set of names introduced in the previous section. The π-calculus *agent terms*, ranged over by $p, q, \ldots$, are closed (wrt. variables X) terms defined by the syntax:

$$p \ ::= \ \mathbf{0} \ \big|\ \pi.p \ \big|\ p|p \ \big|\ p{+}p \ \big|\ (\boldsymbol{\nu} x)\, p \ \big|\ [x{=}y]p \ \big|\ \mathtt{rec}\, X.\, p \ \big|\ X$$

where recursion is guarded[1], and *prefixes*, ranged over by π, are defined as:

$$\pi \ ::= \ \tau \ \big|\ \bar{x}y \ \big|\ x(y).$$

The occurrences of y in $x(y).p$ and $(\boldsymbol{\nu} y)\, p$ are bound; *free names* and *bound names* of agent term p are defined as usual and we denote them with $\mathrm{fn}(p)$ and $\mathrm{bn}(p)$, respectively. Also, we denote with $\mathrm{n}(p)$ and $\mathrm{n}(\pi)$ the sets of (free and bound) names of agent term p and prefix π respectively.

If σ is a name substitution, we denote with $\sigma(p)$ the agent term p whose free names have been replaced according to substitution σ, in a capture-free way.

The *static* π-calculus agent terms, ranged over by s, are defined by the syntax:

$$s \ ::= \ \pi.p \ \big|\ p{+}p \ \big|\ [x{=}y]p \ \big|\ \mathtt{rec}\, X.\, p.$$

We define π-calculus *agents* (π-*agents* in brief) as agent terms up to a *structural congruence* $\equiv$; it is the smallest congruence that satisfies the following axioms:

(alpha)	$p \equiv q$ if p and q are alpha equivalent
(par)	$p\vert\mathbf{0} \equiv p \qquad p\vert q \equiv q\vert p \qquad p\vert(q\vert r) \equiv (p\vert q)\vert r$
(res)	$(\boldsymbol{\nu} x)\,(\boldsymbol{\nu} y)\, p \equiv (\boldsymbol{\nu} y)\,(\boldsymbol{\nu} x)\, p \qquad (\boldsymbol{\nu} x)\, p \equiv p$ if $x \notin \mathrm{fn}(p)$
	$(\boldsymbol{\nu} x)\,(p\vert q) \equiv p\vert(\boldsymbol{\nu} x)\, q$ if $x \notin \mathrm{fn}(p)$

[1] Recursion is guarded in p iff in every subterm $\mathtt{rec}\, X.\, q$ of p, variable X appears within a context $\pi.\_$.

Note that $(\boldsymbol{\nu} x)\,\mathbf{0} \equiv \mathbf{0}$ is a particular case of the last axiom above. We do not consider axioms for summation $(+)$, matching $(=)$, and recursion ($\mathtt{rec}\,.$), since, in this paper, we aim at introducing an algebra with only parallel composition and restriction. We remark that $p \equiv q$ implies $\sigma(p) \equiv \sigma(q)$ and $\mathrm{fn}(p) = \mathrm{fn}(q)$; so, it is possible to define substitutions and free names also for agents.

Below, we introduce an example of a static π-agent. Throughout the paper, we will adopt it as a running example.

Example 1. Let $p = \mathtt{rec}\,X.\,(\boldsymbol{\nu} y)\,\overline{y}x_2.\mathbf{0} \,|\, \overline{x}_2 y.X$ be a static π-agent. Agent p generates new names and extrudes them on a channel x_2; in parallel, p sends name x_2 on each generated channel.

The *actions* an agent can perform are defined by the following syntax:

$$\alpha \ ::= \ \tau \ \big|\ xy \ \big|\ x(z) \ \big|\ \bar{x}y \ \big|\ \bar{x}(z)$$

and are called respectively *synchronization*, *free input*, *bound input*, *free output* and *bound output* actions; x and y are free names of α ($\mathrm{fn}(\alpha)$), whereas z is a bound name ($\mathrm{bn}(\alpha)$); moreover $\mathrm{n}(\alpha) = \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$.

The standard operational semantics of the π-calculus is defined via labeled transitions $p \xrightarrow{\alpha} p'$, where p is the starting agent, p' is the target one and α is an action. We refer to [8] for further explanations of the transition relation.

The operational semantics we consider in this paper is reported in Table 1. It differs from the (standard) *early operational semantics* because it includes a bound input rule (π-INP$'$) and rule (π-CLOSE) replaces rule

$$(\pi\text{-STD-CLOSE})\ \frac{p_1 \xrightarrow{\bar{x}(y)} q_1 \quad p_2 \xrightarrow{xy} q_2}{p_1|p_2 \xrightarrow{\tau} (\boldsymbol{\nu} y)\,(q_1|q_2)} \quad \text{if } y \notin \mathrm{fn}(p_2)$$

To understand why the two semantics are equivalent, let us consider the π-agents $p_1 = (\boldsymbol{\nu} y)\,\overline{x}y.\mathbf{0}$ and $p_2 = \overline{z}y.\mathbf{0} \,|\, x\,(z).\mathbf{0}$. A transition $p_1|p_2 \xrightarrow{\tau} (\boldsymbol{\nu} y)\,\overline{z}y.\mathbf{0}$ should not be allowed because it would cause name y in the first component $\overline{z}y.\mathbf{0}$ of p_2 to be captured by restriction $(\boldsymbol{\nu} y)$. Indeed, on the one hand, if we start with $p_1 \xrightarrow{\overline{x}(y)} \mathbf{0}$ and $p_2 \xrightarrow{x\,y} \overline{z}y.\mathbf{0}$, the above transition is prevented by the side condition of rule (π-STD-CLOSE); on the other hand, if by rule (π-INP) we get $p_2 \xrightarrow{x\,y} \overline{z}y.\mathbf{0}$, rule ($\pi$-CLOSE) cannot be applied since the input action is not bound, while if we employ rule (π-INP$'$) and we obtain $x\,(y).\mathbf{0} \xrightarrow{x\,(y)} \mathbf{0}$, then rule ($\pi$-PAR) cannot be applied to p_2, since y is both bound in $x\,(y)$ and free in $\overline{z}\,y.\mathbf{0}$.

2.3 Coalgebras

We recall that an algebra A over a signature Σ (Σ-algebra in brief) is defined by a carrier set $|A|$ and, for each operation $op \in \Sigma$ of arity n, by a function $op^A : |A|^n \to |A|$. A homomorphism (or simply a morphism) between two Σ-algebras A and B is a function $h : |A| \to |B|$ that commutes with all the operations in Σ, namely, for each operator $op \in \Sigma$ of arity n, we have $op^B(h(a_1), \cdots, h(a_n)) =$

(π-Tau) $\tau.p \xrightarrow{\tau} p$	(π-Out) $\bar{x}y.p \xrightarrow{\bar{x}y} p$
(π-Inp) $x(y).p \xrightarrow{xz} [z \mapsto y]p$	(π-Inp$'$) $x(y).p \xrightarrow{x(y)} p$
(π-Sum) $\dfrac{p_1 \xrightarrow{\alpha} q_1}{p_1{+}p_2 \xrightarrow{\alpha} q_1}$ and symmetric	(π-Match) $\dfrac{p \xrightarrow{\alpha} q}{[x=x]p \xrightarrow{\alpha} q}$
(π-Rec) $\dfrac{p[\mathtt{rec}\,X.\,p/X] \xrightarrow{\alpha} q}{\mathtt{rec}\,X.\,p \xrightarrow{\alpha} q}$	(π-Par) $\dfrac{p \xrightarrow{\alpha} q}{p\|r \xrightarrow{\alpha} q\|r}$ if $\mathrm{bn}(\alpha) \cap \mathrm{fn}(r) = \emptyset$
(π-Res) $\dfrac{p \xrightarrow{\alpha} q}{(\boldsymbol{\nu} y)\,p \xrightarrow{\alpha} (\boldsymbol{\nu} y)\,q}$ if $y \notin \mathrm{n}(\alpha)$	(π-Open) $\dfrac{p \xrightarrow{\bar{x}y} q}{(\boldsymbol{\nu} y)\,p \xrightarrow{\bar{x}(y)} q}$ if $x \neq y$
(π-Com) $\dfrac{p_1 \xrightarrow{\bar{x}y} q_1 \quad p_2 \xrightarrow{xy} q_2}{p_1\|p_2 \xrightarrow{\tau} q_1\|q_2}$	(π-Close) $\dfrac{p_1 \xrightarrow{\bar{x}(y)} q_1 \quad p_2 \xrightarrow{x(y)} q_2}{p_1\|p_2 \xrightarrow{\tau} (\boldsymbol{\nu} y)\,(q_1\|q_2)}$

Table 1. Early operational semantics.

$h(op^A(a_1,\ldots,a_n))$. We denote by $\mathbf{Alg}(\Sigma)$ the category of *Σ-algebras* and *Σ-morphisms*. The following definition introduces labeled transition systems whose states have an algebraic structure.

Definition 1 (transition systems). *Let Σ be a signature, and L be a set of labels. A* transition system *over Σ and L is a pair $lts = \langle A, \longrightarrow_{lts}\rangle$ where A is a nonempty Σ-algebra and $\longrightarrow_{lts} \subseteq |A| \times L \times |A|$ is a labeled transition relation. For $\langle p, l, q\rangle \in \longrightarrow_{lts}$ we write $p \xrightarrow{l}_{lts} q$.*

Let $lts = \langle A, \longrightarrow_{lts}\rangle$ and $lts' = \langle B, \longrightarrow_{lts'}\rangle$ be two transition systems. A morphism $h : lts \to lts'$ of transition systems over Σ and L (lts morphism, in brief) is a Σ-morphism $h : A \to B$ such that $p \xrightarrow{l}_{lts} q$ implies $h(p) \xrightarrow{l}_{lts'} h(q)$.

The notion of bisimulation on structured transition systems is the classical one.

Definition 2 (bisimulation). *Let Σ be a signature, L be a set of labels, and $lts = \langle A, \longrightarrow_{lts}\rangle$ be a transition system over Σ and L.*

A relation $\mathcal{R}$ over $|A|$ is a bisimulation *if $p\,\mathcal{R}\,q$ implies:*

- *for each $p \xrightarrow{l} p'$ there is some $q \xrightarrow{l} q'$ such that $p'\,\mathcal{R}\,q'$;*
- *for each $q \xrightarrow{l} q'$ there is some $p \xrightarrow{l} p'$ such that $p'\,\mathcal{R}\,q'$.*

Bisimilarity *$\sim_{lts}$ is the largest bisimulation.*

Given a signature Σ and a set of labels L, a collection of SOS rules can be regarded as a specification of those transition systems over Σ and L that have a transition relation closed under the given rules.

Definition 3 (SOS rules). *Given a signature Σ and a set of labels L, a sequent $p \xrightarrow{l} q$ (over L and Σ) is a triple where $l \in L$ is a label and p, q are Σ-terms with variables in a given set X.*

An SOS rule r *over* Σ *and* L *takes the form:*

$$\frac{p_1 \xrightarrow{l_1} q_1 \quad \cdots \quad p_n \xrightarrow{l_n} q_n}{p \xrightarrow{l} q}$$

where $p_i \xrightarrow{l_i} q_i$ *as well as* $p \xrightarrow{l} q$ *are sequents.*

We say that transition system $lts = \langle A, \longrightarrow_{lts} \rangle$ satisfies *a rule* r *like above if each assignment to the variables in* X *that is a solution*[2] *to* $p_i \xrightarrow{l_i} q_i$ *for* $i = 1, \ldots, n$ *is also a solution to* $p \xrightarrow{l} q$.

We represent with

$$\frac{\dfrac{p_1 \xrightarrow{l_1} q_1 \quad \cdots \quad p_n \xrightarrow{l_n} q_n}{\vdots}}{p \xrightarrow{l} q}$$

a proof*, with premises* $p_i \xrightarrow{l_i} q_i$ *for* $i = 1, \ldots, n$ *and conclusion* $p \xrightarrow{l} q$*, obtained by applying the rules in* R.

Definition 4 (transition specifications). *A* transition specification *is a tuple* $\Delta = \langle \Sigma, L, R \rangle$ *consisting of a signature* Σ*, a set of labels* L*, and a set of SOS rules* R *over* Σ *and* L.

A transition system over Δ *is a transition system over* Σ *and* L *that satisfies rules* R.

It is well known that ordinary labeled transition systems (i.e., transition systems whose states do not have an algebraic structure) can be represented as coalgebras for a suitable functor [13].

Definition 5 (coalgebras). *Let* $F : \mathcal{C} \to \mathcal{C}$ *be a functor on a category* $\mathcal{C}$*. A* coalgebra for F*, or* F-coalgebra*, is a pair* $\langle A, f \rangle$ *where* A *is an object and* $f : A \to F(A)$ *is an arrow of* $\mathcal{C}$*. A* F-cohomomorphism *(or simply* F-morphism*)* $h : \langle A, f \rangle \to \langle B, g \rangle$ *is an arrow* $h : A \to B$ *of* $\mathcal{C}$ *such that*

$$h; g = f; F(h). \tag{2}$$

We denote with $\mathbf{Coalg}(F)$ *the category of* F*-coalgebras and* F*-morphisms.*

Proposition 1. *For a fixed set of labels* L*, let* $P_L : \mathbf{Set} \to \mathbf{Set}$ *be the functor defined on objects as* $P_L(X) = \mathcal{P}(L \times X + X)$*, where* $\mathcal{P}$ *denotes the countable powerset functor, and on arrows as* $P_L(h)(S) = \{\langle l, h(p) \rangle \mid \langle l, p \rangle \in S \cap L \times X\} \cup \{h(p) \mid p \in S \cap X\}$*, for* $h : X \to Y$ *and* $S \subseteq L \times X + X$*. Then* P_L*-coalgebras are in a one-to-one correspondence with transition systems*[3] *on* L*, given by* $f_{lts}(p) = \{\langle l, q \rangle \mid p \xrightarrow{l}_{lts} q\} \cup \{p\}$ *and, conversely, by* $p \xrightarrow{l}_{lts_f} q$ *if and only if* $\langle l, q \rangle \in f(p)$.

---

[2] Given $h : X \to A$ and its extension $\hat{h} : T_\Sigma(X) \to A$, h is a solution to $p \xrightarrow{l} q$ for lts if and only if $\hat{h}(p) \xrightarrow{l}_{lts} \hat{h}(q)$.

[3] Notice that this correspondence is well defined also for transition systems with sets of states, rather than with algebras of states as required in Definition 1.

In [1] the generalized notion of lax cohomomorphism is given, in order to accommodate also the more general definition of lts morphisms in a (lax) coalgebraic framework. To make clear their intuition, let $f : A \to P_L(A)$ and $g : B \to P_L(B)$ be two P_L-coalgebras and let $h : A \to B$ be a P_L-morphism. If we split the morphism condition (2) for h in the conjunction of the two inclusions $f; P_L(h) \subseteq h; g$ and $h; g \subseteq f; P_L(h)$, then it is easily shown that the first inclusion expresses "preservation" of transitions, while the second one corresponds to "reflection". Thus, lts morphisms can be seen as arrows (i.e., functions in **Set**) that satisfy the first inclusion, while lts morphisms which also satisfy the reflection inclusion are P_L-morphisms. This observation will be useful in Sect. 5.

Definition 6 (De Simone format). *Given a signature Σ and a set of labels L, a rule r over Σ and L is in* De Simone format *if it has the form:*

$$\frac{\{x_i \xrightarrow{l_i} y_i \mid i \in I\}}{op(x_1, \ldots, x_n) \xrightarrow{l} p}$$

where $op \in \Sigma$, $I \subseteq \{1, \ldots, n\}$, p is linear and the variables y_i occurring in p are distinct from variables x_i, except for $y_i = x_i$ if $i \notin I$.

The following results are due to [14] and concern *bialgebras*, i.e., coalgebras in $\mathbf{Alg}(\Sigma)$. Bialgebras enjoy the property that the unique morphism to the final bialgebra, which exists under reasonable conditions, induces a bisimulation that is a congruence with respect to the operations, as noted in the introduction.

Proposition 2 (lifting of P_L). *Let $\Delta = \langle \Sigma, L, R\rangle$ be a transition specification with rules in De Simone format.*

Define $P_\Delta : \mathbf{Alg}(\Sigma) \to \mathbf{Alg}(\Sigma)$ as follows:

- $|P_\Delta(A)| = P_L(|A|)$;
- *whenever* $\dfrac{\{x_i \xrightarrow{l_i} y_i \mid i \in I\}}{op(x_1, \ldots, x_n) \xrightarrow{l} p} \in R$ *then*
  $$\frac{\langle l_i, p_i\rangle \in S_i,\ i \in I \quad q_j \in S_j,\ j \notin I}{\langle l, p[p_i/y_i,\ i \in I,\ \ q_j/y_j,\ j \notin I]\rangle \in op^{P_\Delta(A)}(S_1, \ldots, S_n)};$$
- *if $h : A \to B$ is a morphism in $\mathbf{Alg}(\Sigma)$ then $P_\Delta(h) : P_\Delta(A) \to P_\Delta(B)$ and* $P_\Delta(h)(S) = \{\, \langle l, h(p)\rangle \mid \langle l, p\rangle \in S \cap (L \times |A|) \,\} \cup \{\, h(p) \mid p \in S \cap |A| \,\}$.

Then P_Δ is a well-defined functor on $\mathbf{Alg}(\Sigma)$.

Corollary 1. *Let $\Delta = \langle \Sigma, L, R\rangle$ be a transition specification with rules R in De Simone format.*

Any morphism $h : f \to g$ in $\mathbf{Coalg}(P_\Delta)$ entails a bisimulation $\sim_h$ on lts_f, that coincides with the kernel of the morphism. Bisimulation $\sim_h$ is a congruence for the operations of the algebra.

Moreover, the category $\mathbf{Coalg}(P_\Delta)$ has a final object. Finally, the kernel of the unique P_Δ-morphism from f to the final object of $\mathbf{Coalg}(P_\Delta)$ is a relation on the states of f which coincides with bisimilarity on lts_f and is a congruence.

The theory described so far accounts for the lifting of the functor from **Set** to $\mathbf{Alg}(\Sigma)$; a further step is needed to lift a P_L-coalgebra to be a P_Δ-coalgebra. Indeed, the above step is obvious in the particular case of $f : A \to P_\Delta(A)$, with $A = T_\Sigma$ and f unique by initiality, namely when A has no structural axioms and no additional constants, and lts_f is the minimal transition system satisfying Δ.

As an example that in general the lifting may not succeed, let us consider the case of the chemical abstract machine *CHAM*. For our purposes, the signature Σ_c of a *CHAM* is defined as:

$$\Sigma_c \ ::= \ \mathbf{0} \ \big| \ \_|\_ \ \big| \ a.\_ \ \big| \ \bar{a}.\_ \ \big| \ \mathtt{redex}_a(\_,\_)$$

and E_c is the set of axioms for commutativity, associativity, *id*, **0** plus an axiom $\mathtt{redex}_a(p,q) = a.p \,|\, \bar{a}.q$. The only reduction rule $\mathtt{redex}_a(p,q) \longrightarrow p \,|\, q$ is in De Simone format. For P_{Δ_c} the usual poweralgebra functor on $\mathbf{Alg}(\Sigma_c)$, the transition system of *CHAM* forms a P_L-coalgebra, but not a P_{Δ_c}-coalgebra [4]. And bisimilarity is not a congruence as, for example, $\mathbf{0} \sim a.p$ but $\bar{a}.p \not\sim a.p \,|\, \bar{a}.p$.

In Sect. 5, we will show how to lift a transition system with structural axioms from $\mathbf{Coalg}(P_L)$ to $\mathbf{Coalg}(P_\Delta)$, under appropriate conditions on the axioms.

3 A Category of Coalgebras for Mobile Calculi

In this section, we define a transition specification Δ_{pr} for mobile calculi with permutations (ρ), parallel composition ($|$), name restriction (ν) and extrusion (δ), and prove that the category $\mathbf{Coalg}(P_{\Delta_{pr}})$ of coalgebras over Δ_{pr} is well-defined. As mentioned in the introduction, operators ν and δ do not have arguments, as the extruded or restricted name is assumed to be always the first one, i.e., x_0.

Definition 7. *Signature Σ_{pr} is defined as follows:*

$$\Sigma_{pr} \ ::= \ \mathbf{0} \ \big| \ \_|\_ \ \big| \ \nu.\,\_ \ \big| \ \delta.\,\_ \ \big| \ \rho\_$$

We adopt the convention that operators have decreasing binding power, in the following order: ρ, $|$, ν and δ. Thus, for example, $\nu.\,\delta.\,\rho p|q$ means $\nu.\,(\delta.\,((\rho p)|q))$.

Operators ρ are generic, finite name permutations, as described in Subsect. 2.1. Operator δ is meant to represent the substitution $[x_i \mapsto x_{i+1}]$, for $i = 0, 1, \ldots$. Of course, this substitution is not finite, but, at least in the case of an ordinary agent p, it replaces a finite number of names, i.e., the free names of p.

We define the set L_{pr} of the labels:

$$L_{pr} = \{\tau,\ x\,y,\ x,\ \overline{x}\,y,\ \overline{x} \mid x, y \in \mathfrak{N}\}. \tag{3}$$

If $l \in L_{pr}$ then $\delta\,(l)$ and $\nu\,(l)$ are the labels obtained from l by respectively applying substitution δ and ν to its names, where $\nu\,(x_{i+1}) = x_i$ and $\delta\,(x_i) =$

[4] Indeed, axiom $\mathtt{redex}_a(p,q) = a.p \,|\, \bar{a}.q$ does not satisfy the "bisimulation" condition required in Theorem 7.

x_{i+1}. Homomorphically, δ and ν are extended to π-agents, with δ and ν inactive on bound names. Note that in $\nu.\, p$, p is a value of a Σ_{pr}-algebra; in $\nu\,(p)$, p is a π-agent and $\nu\,(p)$ is defined only if $x_0 \notin \mathrm{fn}(p)$.

The correspondence between labels L_{pr} and the actions of the π-calculus is the obvious one for τ, $x\,y$, and $\overline{x}\,y$. In the case of bound output transitions, only the channel x on which the output occurs is observed in label $\overline{x}$, and similarly in the case of bound input transitions.

Definition 8 (transition specification Δ_{pr}). *The transition specification Δ_{pr} is the tuple $\langle \Sigma_{pr}, L_{pr}, R_{pr} \rangle$, where the signature Σ_{pr} is as in Definition 7, labels L_{pr} are defined in (3) and the rules R_{pr} are the SOS rules in Table 2.*

The most interesting rules are those in the second column, where α is an extrusion or the input of a fresh name. Intuitively, they follow the idea that substitutions on the source of a transition must be reflected on its destination by restoring the extruded or fresh name to x_0. Rule (Delta$'$) applies δ to q and then permutes x_0 and x_1, in order to have the extruded name back to x_0. Conversely, rule (Res$'$) permutes x_0 and x_1 to make sure that the restriction operation applies to x_0 and not to the extruded name x_1. In rule (Par$'$) side condition $\mathrm{bn}(\alpha) \cap \mathrm{fn}(r) = \emptyset$ is *not necessary.* The intuitive reason is that δ shifts all the variables in r to the right and, thus, x_0 does not appear in $\delta.\, r$.

(Rho)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha \neq \overline{x}, x}{\rho p \xRightarrow{\rho(\alpha)} \rho q}$	(Rho$'$)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha = \overline{x}, x}{\rho p \xRightarrow{\rho(\alpha)} \rho_{+1} q}$
(Par)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha \neq \overline{x}, x}{p\lvert r \xRightarrow{\alpha} q\lvert r}$	(Par$'$)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha = \overline{x}, x}{p\lvert r \xRightarrow{\alpha} q\lvert \delta.\, r}$
(Delta)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha \neq \overline{x}, x}{\delta.\, p \xRightarrow{\delta\,(\alpha)} \delta.\, q}$	(Delta$'$)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha = \overline{x}, x}{\delta.\, p \xRightarrow{\delta\,(\alpha)} [x_0 \leftrightarrow x_1]\delta.\, q}$
(Res)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha \neq \overline{x}, x}{\nu.\, p \xRightarrow{\nu\,(\alpha)} \nu.\, q}$ if $x_0 \notin \mathrm{n}(\alpha)$	(Res$'$)	$\dfrac{p \xRightarrow{\alpha} q \quad \alpha = \overline{x}, x}{\nu.\, p \xRightarrow{\nu\,(\alpha)} \nu.\, [x_0 \leftrightarrow x_1] q}$ if $x_0 \notin \mathrm{n}(\alpha)$
(Open)	$\dfrac{p \xRightarrow{\overline{x}\, x_0} q \quad x \neq x_0}{\nu.\, p \xRightarrow{\nu\,(\overline{x})} q}$	(Close)	$\dfrac{p_1 \xRightarrow{\overline{x}} q_1 \quad p_2 \xRightarrow{x} q_2}{p_1\lvert p_2 \xRightarrow{\tau} \nu.\, q_1\lvert q_2}$
(Com)	$\dfrac{p_1 \xRightarrow{\overline{x}\, y} q_1 \quad p_2 \xRightarrow{x\, y} q_2}{p_1\lvert p_2 \xRightarrow{\tau} q_1\lvert q_2}$		

Table 2. Structural operational semantics.

Proposition 3. *Let $\Delta_{pr} = \langle \Sigma_{pr}, L_{pr}, R_{pr} \rangle$ be the transition specification in Definition 8. Rules R_{pr} are in De Simone format.*

Note that the rules of the π-calculus in Table 1 are not in De Simone format: the side condition on $\mathrm{fn}(r)$ prevents rule (π-Par) from that.

Theorem 1. *Let $\Delta_{pr} = \langle \Sigma_{pr}, L_{pr}, R_{pr} \rangle$ be the transition specification in Definition 8. Then, functor $P_{L_{pr}}$ in* **Set** *can be lifted to functor $P_{\Delta_{pr}}$ in $\mathbf{Alg}(\Sigma_{pr})$ and category $\mathbf{Coalg}(P_{\Delta_{pr}})$ is well defined.*

Proof. The claim follows by Proposition 3 and Proposition 2.

4 A Σ_{pr}-Algebra for the π-Calculus

In this section we introduce a Σ_{pr}-algebra for the π-calculus, and we prove a finiteness result.

4.1 A Σ_{pr}-Algebra

We now define a Σ_{pr}-algebra and a bijective translation of π-agents to values of such an algebra.

Definition 9 (Σ_{pr}-algebra for the π-calculus). *Algebra B_π is defined as the initial algebra $B_\pi = T_{(\Sigma_{pr} \cup C_\pi, E_{pr} \cup E')}$ where:*

- *Σ_{pr} is as in Definition 7,*
- *constants C_π are*
$$C_\pi = \{l_s \mid s \text{ is a static } \pi\text{-agent}\}$$
- *axioms E_{pr} are the axioms below:*

$$
\begin{array}{lllll}
(\mathbf{perm}) & (\rho \circ \rho')p = \rho(\rho'(p)) & \mathrm{id}\, p = p & & \\
(\mathbf{par}) & p|\mathbf{0} = p & p|q = q|p & p|(q|r) = (p|q)|r & \\
(\mathbf{res}) & \nu.\, \mathbf{0} = \mathbf{0} & \nu.\, (\delta.\, p)|q = p|\nu.\, q & \nu.\, \nu.\, [x_0 \leftrightarrow x_1]p = \nu.\, \nu.\, p & \\
(\mathbf{delta}) & \delta.\, \mathbf{0} = \mathbf{0} & \delta.\, p \,|\, q = (\delta.\, p) \,|\, \delta.\, q & \delta.\, \nu.\, p = \nu.\, [x_0 \leftrightarrow x_1]\delta.\, p & \\
(\mathbf{rho}) & \rho\, \mathbf{0} = \mathbf{0} & \rho(p|q) = \rho p \,|\, \rho q & \rho\nu.\, p = \nu.\, \rho_{+1} p & \delta.\, \rho p = \rho_{+1}\delta.\, p
\end{array}
$$

- *axioms E' are*
$$\rho l_s = l_{\rho(s)} \qquad \delta.\, l_s = l_{\delta\,(s)}.$$

Axioms (**par**) and (**res**) correspond to the analogous axioms for the π-calculus. The other axioms rule how to invert the order of operators among each other, following the intuition that ν and δ decrease and increase variable indexes, respectively. Axioms E_{pr} and E' can be applied from left to right to reduce every term p into a canonical form $p'(l_{s_1}, \ldots, l_{s_n})$, such that ρ and δ do not occur in context $p'(\_, \_, \ldots)$. Notice that other expected properties like $\nu.\, \delta.\, p = p$ and $[x_0 \leftrightarrow x_1]\delta.\, \delta.\, p = \delta.\, \delta.\, p$ can be derived from these axioms.

Definition 10 (translation $[\![\cdot]\!]$). *We define a translation on π-calculus agent terms $[\![\cdot]\!] : \Pi \to |B|$ as follows:*

$$[\![\mathbf{0}]\!] = \mathbf{0} \quad [\![p|q]\!] = [\![p]\!]|[\![q]\!] \quad [\![(\boldsymbol{\nu} y)\, p]\!] = \nu.\, [\delta\,(y) \leftrightarrow x_0]\delta.\, [\![p]\!] \quad [\![s]\!] = l_s.$$

The translation of the restriction gives the flavor of the De Brujin notation: the idea is to split standard restriction in three steps. First, one shifts all names upwards to generate a fresh name x_0, then swaps $\delta(y)$ and x_0, and, finally, applies restriction on x_0, which now stands for what 'used to be' y.

Translation $[\![\cdot]\!]$ is also defined on actions as follows: $[\![\alpha]\!] = \alpha$, if $\alpha \neq \overline{x}(y), x(y)$; $[\![\overline{x}(y)]\!] = \overline{x}$; $[\![x(y)]\!] = x$.

Theorem 2. *If $p \equiv q$ then $[\![p]\!] = [\![q]\!]$.*

Proof. See the Appendix.

Theorem 3. *Function $[\![\cdot]\!]$ is bijective. I.e., let $\{\![\cdot]\!\} : |B| \to \Pi$ be a translation defined as follows: $\{\![\mathbf{0}]\!\} = \mathbf{0}$; $\{\![p|q]\!\} = \{\![p]\!\}|\{\![q]\!\}$; $\{\![\nu.\, p]\!\} = (\boldsymbol{\nu} x_i)\, \nu\,([\delta(x_i) \leftrightarrow x_0]\{\![p]\!\})$, if $\delta(x_i) \notin \mathrm{fn}(\{\![p]\!\})$; $\{\![l_s]\!\} = s$; $\{\![\rho p]\!\} = \rho(\{\![p]\!\})$; $\{\![\delta.\, p]\!\} = \delta(\{\![p]\!\})$. Then, for every p in B, $[\![\{\![p]\!\}]\!] = p$ and, for every π-agent q, $\{\![[\![q]\!]]\!\} = q$.*

Proof. See the Appendix.

Definition 11 (transition system lts_g). *The transition system for algebra B_π is $lts_g = \langle B_\pi, \Longrightarrow_g\rangle$, where $\Longrightarrow_g$ is defined by the SOS rules in Table 2 plus the following axioms:*

$$\text{(STATIC)}\ \frac{s \xrightarrow{\alpha} r \quad \alpha \neq \overline{x}(y), x(y)}{l_s \overset{[\![\alpha]\!]}{\Longrightarrow}_g [\![r]\!]} \qquad \text{(STATIC}'\text{)}\ \frac{s \xrightarrow{\alpha} r \quad \alpha = \overline{x}(y), x(y)}{l_s \overset{[\![\alpha]\!]}{\Longrightarrow}_g [\delta(y) \leftrightarrow x_0]\delta.\, [\![r]\!]}$$

When no confusion arises, we will simply denote lts_g with lts and $\Longrightarrow_g$ with $\Longrightarrow$.

Theorem 4. *Transition system lts satisfies the specification Δ_{pr} in Definition 8.*

Proof. Trivial, as lts rules include rules R_{pr}.

Example 2. Let us consider again π-agent $p = \mathtt{rec}\, X.\, (\boldsymbol{\nu} y)\, \overline{y}x_2.\mathbf{0}|\overline{x}_2 y.X$. By rule ($\pi$-REC) and ($\pi$-OPEN), p can reduce as $p \xrightarrow{\overline{x}_2(x_3)} \overline{x}_3 x_2.\mathbf{0}\,|\,p$. On the other hand, $[\![p]\!] = l_p$. By rule (STATIC′), l_p can reduce as $l_p \overset{\overline{x}_2}{\Longrightarrow} [\delta(x_3) \leftrightarrow x_0]\delta.\, [\![\overline{x}_3 x_2.\mathbf{0}\,|\,p]\!]$ and $[\delta(x_3) \leftrightarrow x_0]\delta.\, [\![\overline{x}_3 x_2.\mathbf{0}\,|\,p]\!] = [\![\overline{x}_0 x_3.\mathbf{0}]\!]\,|\,[x_4 \leftrightarrow x_0]\delta.\, l_p$.

At a second step, by rule (π-REC) and (π-OPEN), p can reduce as $p \xrightarrow{\overline{x}_2(x_1)} \overline{x}_1 x_2.\mathbf{0}\,|\,p$, and by rule ($\pi$-PAR) $\overline{x}_3 x_2.\mathbf{0}\,|\,p \xrightarrow{\overline{x}_2(x_1)} \overline{x}_3 x_2.\mathbf{0}\,|\,\overline{x}_1 x_2.\mathbf{0}\,|\,p$. On the other hand, $[\![\overline{x}_0 x_3.\mathbf{0}]\!]\,|\,[x_4 \leftrightarrow x_0]\delta.\, l_p \overset{\overline{x}_3}{\Longrightarrow} [\![\overline{x}_1 x_4.\mathbf{0}]\!]\,|\,[\![\overline{x}_0 x_4.\mathbf{0}]\!]\,|\,\delta.\,\delta.\, l_p$. Indeed, by rule (STATIC′), $l_p \overset{\overline{x}_2}{\Longrightarrow} [x_2 \leftrightarrow x_0]\delta.\, [\![\overline{x}_1 x_2.\mathbf{0}\,|\,p]\!]$ and $[x_2 \leftrightarrow x_0]\delta.\, [\![\overline{x}_1 x_2.\mathbf{0}\,|\,p]\!] = [\![\overline{x}_0 x_3.\mathbf{0}]\!]\,|\,[x_2 \leftrightarrow x_0]\delta.\, l_p$. Then, by (DELTA′), $\delta.\, l_p \overset{\overline{x}_3}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, [\![\overline{x}_0 x_3.\mathbf{0}]\!]\,|\,\delta.\, l_p = [\![\overline{x}_0 x_4.\mathbf{0}]\!]\,|\,\delta.\,\delta.\, l_p$. By rule (RHO′), $[x_4 \leftrightarrow x_0]\delta.\, l_p \overset{\overline{x}_3}{\Longrightarrow} [x_5 \leftrightarrow x_1][\![\overline{x}_0 x_4.\mathbf{0}]\!]\,|\,\delta.\,\delta.\, l_p = [\![\overline{x}_0 x_4.\mathbf{0}]\!]\,|\,\delta.\,\delta.\, l_p$. Finally, $[\![\overline{x}_0 x_3.\mathbf{0}]\!]\,|\,[x_4 \leftrightarrow x_0]\delta.\, l_p \overset{\overline{x}_3}{\Longrightarrow} [\![\overline{x}_1 x_4.\mathbf{0}]\!]\,|\,[\![\overline{x}_0 x_4.\mathbf{0}]\!]\,|\,\delta.\,\delta.\, l_p$, by rule (PAR′).

The two lemmata below will be useful to prove that the transition system of the π-calculus is equivalent to $lts = \langle B_\pi, \Longrightarrow \rangle$.

Lemma 1. *Let p and q be two π-agents. If $p \sim q$, then $\delta(p) \sim \delta(q)$ and $\rho(p) \sim \rho(q)$.*

Lemma 2.

1. *Let p and q be two π-agents. If $p \xrightarrow{\alpha} q$ and $\alpha \neq \overline{x}(y), x(y)$ then $[\![p]\!] \xRightarrow{[\![\alpha]\!]} [\![q]\!]$; if $p \xrightarrow{\alpha} q$ and $\alpha = \overline{x}(y), x(y)$ then $[\![p]\!] \xRightarrow{[\![\alpha]\!]} [\delta(y) \leftrightarrow x_0]\delta.[\![q]\!]$.*
2. *Let p and q be in B. If $p \xRightarrow{\alpha} q$ and $\alpha \neq \overline{x}, x$, then $\{\![p]\!\} \xrightarrow{\alpha} \{\![q]\!\}$; if $p \xRightarrow{\overline{x}} q$ (resp. $p \xRightarrow{x} q$), then $\{\![p]\!\} \xrightarrow{\overline{x}(x_i)} \nu([\delta(x_i) \leftrightarrow x_0]\{\![q]\!\})$, (resp. $\{\![p]\!\} \xrightarrow{x(x_i)} \nu([\delta(x_i) \leftrightarrow x_0]\{\![q]\!\})$), for every x_i such that $\delta(x_i) \notin \mathrm{fn}(\{\![q]\!\})$.*

Proof. See the Appendix.

4.2 A Finiteness Result

We now prove that, for each π-agent p, the set of static agents and associated reduction rules needed in all derivations of p is finite up to name permutations. It is not larger than the set of the static subagents of p, closed with respect to name fusions.

Definition 12.

- *Let s be a static π-agent. We define $S(s)$ as follows: $S(q_1 + q_2) = \mathcal{S}([\![q_1]\!]) \cup \mathcal{S}([\![q_2]\!]) \cup \{q_1 + q_2\}$; $S(\mathtt{rec}\,X.\,q) = [\mathtt{rec}\,X.\,q \mapsto X]\mathcal{S}([\![q]\!]) \cup \{\mathtt{rec}\,X.\,q\}$; $S(X) = \emptyset$; $S([x = y]q) = \mathcal{S}([\![q]\!]) \cup \{[x = y]q\}$.*
- *Let p be a term of B_π in the canonical form $p(l_{s_1}, \ldots, l_{s_n})$ such that ρ and δ do not occur in context $p(\_, \_, \ldots)$. We define $\mathcal{S}(p)$ as follows: $\mathcal{S}(\nu.\,p) = \mathcal{S}(p)$; $\mathcal{S}(p|q) = \mathcal{S}(p) \cup \mathcal{S}(q)$; $\mathcal{S}(\mathbf{0}) = \emptyset$; $\mathcal{S}(l_s) = S(s)$.*

Notice that $S(s)$ is defined only for static agents, as, for the subagents q of s, $\mathcal{S}([\![q]\!])$ is recalled.

Lemma 3. *Let p and q be terms of B_π. If $p \xRightarrow{\alpha} q$ then for each $q' \in \mathcal{S}(q)$ there is a $p' \in \mathcal{S}(p)$ with $q' = \sigma p'$, for some name substitution σ (not necessarily a permutation).*

Theorem 5. *Let p be in B_π. The number of constants l_s (and respective axioms for $\Longrightarrow$) in all derivations of p is finite, up to name permutations.*

Proof. Set $\mathcal{S}(p)$ is finite for every p, since it is defined by structural recursion. Thus, also its closure with respect to name fusion $\hat{\mathcal{S}}(p) = \{\sigma q \mid q \in \mathcal{S}(p),\ \sigma \text{ name substitution}\}$ is finite, when taken up to name permutations. But Lemma 3 guarantees that $p \xRightarrow{\alpha} q$ implies $\hat{\mathcal{S}}(q) \subseteq \hat{\mathcal{S}}(p)$.

The condition "up to name permutations" is not restrictive. Indeed, the idea is that, for each agent p, we can give a finite number of clauses $l_q \stackrel{\alpha}{\Longrightarrow} q'$ such that the transitions of static agents in all the derivations of p only use permutations of the form $l_{\rho q} \stackrel{\rho(\alpha)}{\Longrightarrow} \rho' q'$, with $\rho' = \rho$ or $\rho' = \rho_{+1}$ according to α.

Example 3. Let us consider again agent $p = \mathtt{rec}\, X.\, q$, with $q = (\boldsymbol{\nu} y)\, \overline{y} x_2.\mathbf{0} \,|\, \overline{x}_2 y.X$. $\mathcal{S}([\![p]\!]) = \{p\} \cup [\mathtt{rec}\, X.\, q \mapsto X]\mathcal{S}([\![q]\!])$. Since $[\![q]\!] = \nu.\, l_{\overline{x}_0 x_3.\mathbf{0}} \,|\, l_{\overline{x}_3 x_0.X}$, then $\mathcal{S}([\![q]\!]) = \{\overline{x}_0 x_3.\mathbf{0}\} \cup \{\overline{x}_3 x_0.X\}$ and $\mathcal{S}([\![p]\!]) = \{p\} \cup \{\overline{x}_0 x_3.\mathbf{0}\} \cup \{\overline{x}_3 x_0.\mathtt{rec}\, X.\, q\}$.

5 Lifting Coalgebras on Set to Coalgebras on Alg(Σ)

In this section we prove a general result about lifting a coalgebra g in **Set** with set of states B and equipped with operations Σ, auxiliary constants C, structural axioms E and De Simone rules R to a coalgebra in $\mathbf{Alg}(\Sigma)$. The lifting allows to apply Corollary 1 and, in particular, to prove that bisimilarity is a congruence.

Theorem 6. *Let $\mathcal{B}$ be the class of coalgebras g in* **Set** *with the following properties:*

1. $g : |B| \to P_L(|B|)$, *with* $B = T_{(\Sigma \cup C, E)}$.
2. lts_g *satisfies transition specification* $\Delta = (\Sigma, L, R)$, *with R in De Simone format.*
3. *A set of basic transitions* $T \subseteq C \times L \times T_{\Sigma \cup C}$ *exists for constants C, namely,* $(c, l, t) \in T$ *implies* $c \xrightarrow{l}_g [t]_E$.

Then, there is an initial coalgebra $\hat{g}$ in $\mathcal{B}$, such that $\forall g \in \mathcal{B}$, $\forall p \in B$, $p \xrightarrow{l}_{\hat{g}} q$ implies $p \xrightarrow{l}_g q$.

Furthermore, the transitions of $\hat{g}$ can be derived using the rules R and the following additional rules:

$$(\textsc{Const})\ \frac{(c,l,t) \in T}{c \xrightarrow{l}_{\hat{g}} t} \qquad\qquad (\textsc{Struct})\ \frac{t_1 =_E t_1' \quad t_1' \xrightarrow{l}_{\hat{g}} t_2' \quad t_2' =_E t_2}{t_1 \xrightarrow{l}_{\hat{g}} t_2}$$

where terms t, t_1, t_1', t_2, t_2' are in $T_{\Sigma \cup C}$.

Proof. Signature $\Sigma \cup C$, axioms E and transition specification Δ can be considered as a single algebraic specification, equipped with an initial model. The elements of the model can be derived using the proof system associated to the specification.

Remark 1. Let A be a Σ-algebra and $h : X \to |A|$ be a function in **Set**. Then, h can be uniquely extended to $\hat{h} : T_\Sigma(X) \to A$ in $\mathbf{Alg}(\Sigma)$. For simplicity, in the sequel, we will also denote $\hat{h}$ by h.

Definition 13. *Let $g : |B| \to P_L(|B|)$ be the initial coalgebra of Theorem 6, where, however, constants C are considered as auxiliary, i.e., B is seen as a Σ-algebra. Then, we define the following Σ-algebras and Σ-morphisms:*

– *$A = T_\Sigma(C)$ and $h : A \to B$ as the unique extension in* $\mathbf{Alg}(\Sigma)$ *of $h(c) = [c]_E$, for $c \in C$;*
– *$f : A \to P_\Delta(A)$ as the unique extension of $f(c) = \{(l,t) \mid (c,l,t) \in T\}$ in* $\mathbf{Alg}(\Sigma)$.

In the sequel, we want to find conditions under which P_L-coalgebra g can be lifted to a P_Δ-coalgebra and function h, as above defined, to a P_Δ-morphism. The observation in Subsect. 2.3 allows us to state, without any further condition, that h is a lax morphism between P_L-coalgebras f and g. The reflection inclusion, instead, will require appropriate hypotheses.

Property 1. Function h in Definition 13 is a lts morphism, namely, a lax P_L-coalgebra morphism. Furthermore, it is surjective.

Proof. Immediate, as every proof of a transition in lts_f holds also in lts_g.

Theorem 7. *Let g be the initial coalgebra in $\mathcal{B}$ as specified by Theorem 6, and let A, h, and f be defined as in Definition 13. Then, h is surjective. Let us assume that for all equations $t_1 = t_2$ in E, with free variables $\{x_i\}_{i \in I}$, we have De Simone proofs as follows:*

$$\frac{x_i \xrightarrow{l_i} y_i \quad i \in I}{t_1 \xrightarrow{l} t_1'} \quad \textit{implies} \quad \frac{x_i \xrightarrow{l_i} y_i \quad i \in I}{t_2 \xrightarrow{l} t_2'} \quad \textit{and} \quad t_1' =_E t_2' \tag{4}$$

and viceversa, using the rules in R and the additional rules:

$$c \xrightarrow{l} t \quad \textit{iff} \quad (c,l,t) \in T.$$

Then, the left diagram below commutes in $\mathbf{Set}$, *i.e., $h; g = f; P_L(h)$. Thus, h is a P_L-morphism.*

Proof. We start noticing that $h : A \to B$ is surjective since $B = A_{/=_E}$ and $=_E$ is the kernel of h. Then, we first prove that the equivalence relation $=_E$ is a bisimulation for lts_f. We use rule induction on the proofs of $=_E$. For axioms in E , the property is guaranteed by Condition 4. Rules for reflexivity, symmetry and transitivity are obviously satisfied. Rule for congruence is also easily checked, since if t_i and t_i' bisimulate, for $i = 1, \cdots, n$, also $k(t_1, \cdots, t_n)$ and $k(t_1', \cdots, t_n')$, with $k \in \Sigma_n$, have corresponding transitions by applying the same De Simone rule or the same constant rule.

Since h is a lax P_L-coalgebra morphism, to derive our result it is enough to show that $h(t_1) \xrightarrow{l}_g p$ implies $t_1 \xrightarrow{l}_f t_2$, with $h(t_2) = p$. We prove this property by induction on the rules of lts_g, as specified by Theorem 6. Rules in R and rule (Const) can be easily checked since they are the same for lts_f and for lts_g. Also, the last rule (Struct) preserves the property. Indeed, given $[t_1']_E \xrightarrow{l}_g [t_2']_E$, i.e., $h(t_1') \xrightarrow{l}_g h(t_2')$, by induction hypothesis we have $t_1' \xrightarrow{l}_f t_2''$, with $h(t_2'') = h(t_2')$. Furthermore, since $t_1 =_E t_1'$ and $=_E$ is a bisimulation for f, we can find $t_1 \xrightarrow{l}_f t_2'''$, with $t_2''' =_E t_2''$ and, thus, $h(t_2''') = h(t_2'')$. Also, $t_2' =_E t_2$ implies $h(t_2) = h(t_2')$. Then, $h(t_2''') = h(t_2)$.

$$\begin{tikzcd} {|A|} \arrow[r, "h"] \arrow[d, "f"'] & {|B|} \arrow[d, "g"] \\ P_L(|A|) \arrow[r, "P_L(h)"'] & P_L(|B|) \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "h"] \arrow[d, "f"'] & B \arrow[d, "g"] \\ P_\Delta(A) \arrow[r, "P_\Delta(h)"'] & P_\Delta(B) \end{tikzcd}$$

Theorem 8. *Let g be the initial coalgebra in $\mathcal{B}$ as specified by Theorem 6, and let A, h, and f be defined as in Definition 13. If the left diagram above commutes in* **Set***, i.e., $h; g = f; P_L(h)$, then g can be lifted from* **Set** *to* $\mathbf{Alg}(\Sigma)$ *and the right diagram commutes in* $\mathbf{Alg}(\Sigma)$.

Proof. See the Appendix.

Corollary 2. *Let g be the initial coalgebra in $\mathcal{B}$ as specified by Theorem 6, and let the right diagram above commute. Then in g bisimilarity is a congruence.*

Proof. The claim follows by Theorem 8 and Corollary 1.

6 A Bialgebra for the π-Calculus

In this section, we apply the results of Sect. 5 to the permutation algebra B_π and to the transition system $\langle B_\pi, \Longrightarrow_g \rangle$. We will prove that axioms E_{pr} satisfy the "bisimulation" condition in Theorem 7 and that $\langle B_\pi, \Longrightarrow_g \rangle$ is equivalent to the transition system of the π-calculus. Thus, bisimilarity is a congruence with respect to parallel composition and restriction.

Theorem 9. *If we consider Δ_{pr} as transition specification, algebra B_π defined in Definition 9, with $E_{pr} \cup E' = E$ and $C_\pi = C$. Then, Condition 4 in Theorem 7 holds.*

Proof. See the Appendix.

Corollary 3. *Let B_π be the algebra defined in Definition 9. Bisimilarity is a congruence in $g : B_\pi \to P_{\Delta_{pr}}(B_\pi)$.*

Proof. The claim derives by Proposition 3, Theorem 9, Theorem 8 and Theorem 7.

Theorem 10. *Let p and q be π-agents. Then, $p \sim q$ if and only if $[\![p]\!] \sim_g [\![q]\!]$.*

Proof. The proof is based on the fact that, by Corollary 3, bisimilarity $\sim_g$ is a congruence with respect to parallel composition and restriction. See the appendix.

Corollary 4. *Bisimilarity $\sim$ in the π-calculus is a congruence with respect to parallel composition and restriction.*

Proof. See the Appendix.

7 Conclusions

In this paper we have presented a general result about lifting calculi with structural axioms to coalgebraic models and we have applied such a result to the π-calculus. We have enriched the permutation algebra for the calculus defined in [9, 10] with parallel composition, restriction and an extrusion operation δ. Since this algebra satisfies the condition required by our general theory (i.e., structural axioms bisimulate), the associated transition system can be lifted to obtain a bialgebra and, as a consequence, in the π-calculus, bisimilarity is a congruence with respect to parallel composition and restriction. To achieve this result, our algebra features no prefix operation: we rely, instead, on a clause format defining the transitions of recursively defined processes. Therefore, the bisimilarity induced by the unique morphism to the final bialgebra is the observational equivalence, rather than observational congruence, as in [6].

We expect that our coalgebraic model can be extended to treat weak bisimulation. Instead, we do not know how to adapt our model to deal with the late bisimilarity equivalence. In [5], late bisimilarity is seen in terms of early bisimulation. The idea is to split a bound input transition in two parts: in the first one, only the channel is observed; in the second one, there are infinitely many transitions, each labeled with a substitution for the input name. While the first transition is easy to model, the second kind of transitions requires a rule employing a substitution which is not a permutation.

We are rather confident that π-calculus observational congruences (both early and late) can be easily handled in our approach by considering algebras of name substitutions rather than name permutations. The same extension should accommodate also calculi with name fusions. More challenging would be to introduce general substitutions (on some first order signature), yielding models rather close to logic programming. By varying the underlying algebra, but otherwise employing similar constructions, we plan to show the flexibility of our uniform bialgebraic approach, where substitutions and extrusion are at the same level as the other operations of the algebra.

References

1. A. Corradini, M. Große-Rhode, R. Heckel. Structured transition systems as lax coalgebras. In *Proc. of CMCS'98*, ENTS 11. Elsevier Science, 1998.
2. A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: A coalgebraic view of open systems. *Theoretical Computer Science* 280:163-192, 2002.
3. R. De Simone. Higher level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science* 37(3):245–267, 1985.
4. G. Ferrari, U. Montanari, and M. Pistore. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. In *Proc. of FoSSaCS'02*, LNCS 2303. Springer, 2002.
5. G. Ferrari, U. Montanari, and P. Quaglia. A pi-calculus with Explicit Substitutions. *Theoretical Computer Science* 168(1):53–103, 1996.
6. M. Fiore and D. Turi. Semantics of name and value passing. In *Proc. of LICS'01*, IEEE. Computer Society Press, 2001.

7. M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *Proc. of LICS'99*, IEEE. Computer Society Press, 1999.
8. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts I and II). *Information and Computation*, 100(1):1–77, 1992.
9. U. Montanari and M. Pistore. Pi-Calculus, Structured Coalgebras and Minimal HD-Automata. In *Proc. of MFCS'00*, LNCS 1983. Springer, 2000.
10. U. Montanari and M. Pistore. Structured Coalgebras and Minimal HD-Automata for the pi-Calculus. Technical Report 0006-02, IRST-ITC, 2000. Available at the URL: `http://sra.itc.it/paper.epl?id=MP00`.
11. A. M. Pitts. Nominal Logic: A First Order Theory of Names and Binding. In *Proc. of TACS'01*, LNCS 2215. Springer, 2001.
12. M. Pistore. *History Dependent Automata*. PhD. Thesis TD-5/99. Università di Pisa, Dipartimento di Informatica, 1999. Available at the URL: `http://www.di.unipi.it/phd/tesi/tesi_1999/TD-5-99.ps.gz`.
13. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science* 249(1):3–80, 2000.
14. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. of LICS'97*, IEEE. Computer Society Press, 1997.

A Proofs

Proof of Theorem 2. The proof is by structural induction on the axioms of the structural congruence $\equiv$.

First, note that $[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![r]\!] = \delta.\,[\![r]\!]$, if $x \notin \mathrm{fn}(r)$. Indeed, it is easily proved by first applying axioms E' and then substitution $[\delta\,(x) \leftrightarrow x_0]$ to the π-agent $\delta\,(p)$.

The most interesting cases of the proof concern alpha equivalent non-static agents and restriction.

Ax. $(\boldsymbol{\nu} x)\,p \equiv (\boldsymbol{\nu} y)\,[y \mapsto x]p,\ y \notin \mathrm{fn}(p)$.
By the condition, $(\boldsymbol{\nu} y)\,[y \mapsto x]p = (\boldsymbol{\nu} y)\,[y \leftrightarrow x]p$. $[\![(\boldsymbol{\nu} y)\,[x \leftrightarrow y]p]\!] = \nu.\,[\delta\,(y) \leftrightarrow x_0]\delta.\,[\![[x \leftrightarrow y]p]\!] = \nu.\,[\![[\delta\,(y) \leftrightarrow x_0][\delta\,(x) \leftrightarrow \delta\,(y)]\delta\,(p)]\!] = \nu.\,[\![[\delta\,(x) \leftrightarrow x_0]\delta\,(p)]\!] = [\![(\boldsymbol{\nu} x)\,p]\!]$.

Ax. $(\boldsymbol{\nu} x)\,(\boldsymbol{\nu} y)\,p \equiv (\boldsymbol{\nu} y)\,(\boldsymbol{\nu} x)\,p$.
First, $[\![(\boldsymbol{\nu} x)\,(\boldsymbol{\nu} y)\,p]\!] = \nu.\,\nu.\,\rho_1\delta\delta[\![p]\!]$ and $[\![(\boldsymbol{\nu} y)\,(\boldsymbol{\nu} x)\,p]\!] = \nu.\,\nu.\,\rho_2\delta\delta[\![p]\!]$, for some ρ_1 and ρ_2 that differ for a permutation $[x_0 \leftrightarrow x_1]$. Then, $[\![(\boldsymbol{\nu} x)\,(\boldsymbol{\nu} y)\,p]\!] = [\![(\boldsymbol{\nu} y)\,(\boldsymbol{\nu} x)\,p]\!]$, by axiom $\nu.\,\nu.\,q = \nu.\,\nu.\,[x_0 \leftrightarrow x_1]q$.

Ax. $(\boldsymbol{\nu} x)\,(p|q) \equiv p|(\boldsymbol{\nu} x)\,q$ if $x \notin \mathrm{fn}(p)$.
$[\![(\boldsymbol{\nu} x)\,(p|q)]\!] = \ldots = \nu.\,([\delta\,(x) \leftrightarrow x_0]\delta.\,[\![p]\!]\,|[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![q]\!])$. Since $x \notin \mathrm{fn}(p)$, $[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![p]\!] = \delta.\,[\![p]\!]$. Then, $[\![(\boldsymbol{\nu} x)\,(p|q)]\!] = \nu.\,(\delta.\,[\![p]\!]|[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![q]\!]) = [\![p]\!]|\nu.\,[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![q]\!] = [\![p|(\boldsymbol{\nu} x)\,q]\!]$.

Ax. $(\boldsymbol{\nu} x)\,p \equiv p$, if $x \notin \mathrm{fn}(p)$.
It follows by the condition, by the observation on top of the proof, and by axiom $\nu.\,(\delta.\,p|q) = p|\nu.\,q$, with $q = 0$.

Proof of Theorem 3. We first prove that, for $q \in B$, $\llbracket \{\![q]\!\} \rrbracket = q$. We just consider the case $q = \nu.\,p$, which is the most interesting case.
$\llbracket \{\![\nu.\,p]\!\} \rrbracket = \llbracket (\boldsymbol{\nu} x_i)\,\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\}) \rrbracket = \nu.\,[\delta\,(x_i) \leftrightarrow x_0]\delta.\,\llbracket \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\}) \rrbracket$ $= \nu.\,\llbracket [\delta\,(x_i) \leftrightarrow x_0]\delta\,(\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\})) \rrbracket$. Since $[\delta\,(x_i) \leftrightarrow x_0]\delta\,(\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\})) = \{\![p]\!\}$ and $\llbracket \{\![p]\!\} \rrbracket = p$, then $\llbracket \{\![\nu.\,p]\!\} \rrbracket = \nu.\,p$. Note that the condition is necessary for the application of ν.

We now prove that, for a π-calculus agent r, $\{\![\llbracket r \rrbracket]\!\} = r$. We just consider the case $r = (\boldsymbol{\nu} x)\,p$, for some x.
$\{\![\llbracket (\boldsymbol{\nu} x)\,p \rrbracket]\!\} = (\boldsymbol{\nu} x)\,\nu\,([\delta\,(x) \leftrightarrow x_0]\{\![[\delta\,(x) \leftrightarrow x_0]\delta.\,\llbracket p \rrbracket]\!\})$, since $x \notin \mathrm{fn}([\delta\,(x) \leftrightarrow x_0]\delta.\,\llbracket p \rrbracket)$. Then, $\{\![\llbracket (\boldsymbol{\nu} x)\,p \rrbracket]\!\} = \ldots = (\boldsymbol{\nu} x)\,\{\![\nu.\,\delta.\,\llbracket p \rrbracket]\!\} = (\boldsymbol{\nu} x)\,p$, by axiom $\nu.\,(\delta.\,p)|q$ $= p|\nu.\,q$, with $q = \mathbf{0}$, and by induction hypothesis.

Proof of Lemma 2. The proof of (1) is by induction on the rules $\rightarrow$. We just consider the most interesting cases.

Rule (π-Res). By the condition, $\alpha \neq \overline{x}\,(y), x\,(y)$. By induction hypothesis, $\llbracket p \rrbracket \stackrel{\llbracket \alpha \rrbracket}{\Longrightarrow} \llbracket q \rrbracket$. It follows that $[\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket p \rrbracket \stackrel{\delta\,(\llbracket \alpha \rrbracket)}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q \rrbracket$, by applying rules $\Rightarrow$ and exploiting that $[\delta\,(y) \leftrightarrow x_0]\delta\,(\llbracket \alpha \rrbracket) = \delta\,(\llbracket \alpha \rrbracket)$, for $y \notin \mathrm{n}(\llbracket \alpha \rrbracket)$. Since $x_0 \notin \delta\,(\llbracket \alpha \rrbracket)$, $\llbracket \alpha \rrbracket \neq \overline{x}, x$, and $\nu\,(\delta\,(\llbracket \alpha \rrbracket)) = \llbracket \alpha \rrbracket$, $\llbracket (\boldsymbol{\nu} y)\,p \rrbracket \stackrel{\llbracket \alpha \rrbracket}{\Longrightarrow} \llbracket (\boldsymbol{\nu} y)\,q \rrbracket$.

Rule (π-Close). By hypothesis $\llbracket p_1 \rrbracket \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q_1 \rrbracket$ and $\llbracket p_2 \rrbracket \stackrel{x}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q_2 \rrbracket$. Then, $\llbracket p_1 | p_2 \rrbracket \stackrel{\tau}{\Longrightarrow} \nu.\,[\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q_1 | q_2 \rrbracket$.

Rule (π-Open). By hypothesis, $\llbracket p \rrbracket \stackrel{\overline{x}\,y}{\Longrightarrow} \llbracket q \rrbracket$. Thus, $[\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket p \rrbracket \stackrel{\delta\,(\overline{x})x_0}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q \rrbracket$, by applying rules $\Rightarrow$ and noting that $[\delta\,(y) \leftrightarrow x_0]\delta\,(\overline{x}\,y) = \delta\,(\bar{x})x_0$. It follows that $\nu.\,[\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket p \rrbracket \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q \rrbracket$, with $\nu\,(\delta\,(\overline{x})) = \overline{x}$.

Rule (π-Par). If $\alpha \neq \overline{x}\,(y), x\,(y)$, the proof is trivial. Suppose $\alpha = \overline{x}\,(y)$ (the proof is similar if $\alpha = \overline{x}\,(y)$). By induction hypothesis, $\llbracket p \rrbracket \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q \rrbracket$. Then, $\llbracket p \rrbracket | \llbracket r \rrbracket \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0](\delta.\,\llbracket q \rrbracket)|\delta.\,\llbracket r \rrbracket$ and $\llbracket p|r \rrbracket \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,\llbracket q|r \rrbracket$, as $y \notin \mathrm{fn}(r)$.

The proof of (2) is by induction on the rules $\Rightarrow$. We just consider the most interesting cases.

Rule (Static$'$). Suppose $\alpha = \overline{x}$ (proof is similar if $\alpha = x$). We apply the rule with $l_s = p$ and $r = \nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q]\!\})$ and exploit that $\llbracket [\delta\,(y) \leftrightarrow x_0]\delta\,(\nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q]\!\})) \rrbracket = \llbracket \{\![q]\!\} \rrbracket$.

Rule (Par$'$). Let $\alpha = \overline{x}$ (similarly, if $\alpha = x$). By hypothesis, $\{\![p]\!\} \stackrel{\overline{x}\,(x_i)}{\longrightarrow} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, for every x_i, such that $\delta\,(x_i) \notin \mathrm{fn}(\{\![q]\!\})$. Under the hypothesis $\delta\,(x_i) \notin \mathrm{fn}(\{\![q]\!\}|\delta\,(\{\![r]\!\}))$, $x_i \notin \mathrm{fn}(\{\![r]\!\})$. Thus, $\{\![p]\!\}|\{\![r]\!\} \stackrel{\overline{x}\,(x_i)}{\longrightarrow} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})|\{\![r]\!\}$ and $\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})|\{\![r]\!\} = \nu\,([\delta\,(x_i) \leftrightarrow x_0](\{\![q]\!\}|\{\![\delta.\,r]\!\}))$.

Rule (Delta$'$). Suppose $\alpha = \overline{x}$. By hypothesis, $\{\![p]\!\} \stackrel{\overline{x}\,(x_i)}{\longrightarrow} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, for every x_i s.t. $\delta\,(x_i) \notin \mathrm{fn}(\{\![q]\!\})$. Thus, $\delta\,(\{\![p]\!\}) \stackrel{\alpha'}{\longrightarrow} \delta\,(\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\}))$, with $\alpha' = \delta\,(\overline{x})(\delta\,(x_i))$, and $\delta\,(\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})) = \nu\,([\delta\,(\delta\,(x_i)) \leftrightarrow x_0]$ $[x_0 \leftrightarrow x_1]\ \delta\,(\{\![q]\!\}))$.

Rule (Close). By hypothesis, $\{\![p_1]\!\} \xrightarrow{\overline{x}\,(y)} \nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q_1]\!\})$, for every y s.t. $\delta\,(y) \notin \mathrm{fn}(\{\![q_1]\!\})$; and $\{\![p_2]\!\} \xrightarrow{x\,(k)} \nu\,([\delta\,(k) \leftrightarrow x_0]\{\![q_2]\!\})$, for every k s.t. $\delta\,(k) \notin \mathrm{fn}(\{\![q_2]\!\})$. Thus, $\{\![p_1]\!\}|\{\![p_2]\!\} \xrightarrow{\tau} (\boldsymbol{\nu} y)\,(\nu\,([\delta\,(y) \leftrightarrow x_0](\{\![q_1]\!\}|\{\![q_2]\!\})))$, by applying rule ($\pi$-Close) with $y = k$, and $(\boldsymbol{\nu} y)\,(\nu\,([\delta\,(y) \leftrightarrow x_0](\{\![q_1]\!\}|\{\![q_2]\!\}))) = \{\![\nu.\, q_1|q_2]\!\}$.

Rule (Res). By hypothesis, $\{\![p]\!\} \xrightarrow{\alpha} \{\![q]\!\}$. We take x_i s.t. $\delta\,(x_i) \notin \mathrm{fn}(\{\![p]\!\}) \cup \mathrm{fn}(\{\![q]\!\}) \cup \mathrm{n}(\alpha)$. Then $\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\}) \xrightarrow{\nu\,(\alpha)} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, exploiting that $\nu\,([\delta\,(x_i) \leftrightarrow x_0]\alpha) = \nu\,(\alpha)$. Since $x_i \notin \mathrm{n}(\nu\,(\alpha))$, by rule ($\pi$-Res), $(\boldsymbol{\nu} x_i)\,\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\}) \xrightarrow{\nu\,(\alpha)} (\boldsymbol{\nu} x_i)\,\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, i.e., $\{\![\nu.\, p]\!\} \xrightarrow{\nu\,(\alpha)} \{\![\nu.\, q]\!\}$.

Rule (Res′). Suppose $\alpha = \overline{x}$ (similar proof, otherwise). By hypothesis, $\{\![p]\!\} \xrightarrow{\overline{x}\,(x_i)} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, with $\delta\,(x_i) \notin \mathrm{fn}(\{\![q]\!\})$.
$(\boldsymbol{\nu} x_k)\,\nu\,([\delta\,(x_k) \leftrightarrow x_0]\{\![p]\!\}) \xrightarrow{\alpha'} (\boldsymbol{\nu} x_k)\,\nu\,([\delta\,(x_k) \leftrightarrow x_0]\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\}))$, with $\alpha' = \nu\,([\delta\,(x_k) \leftrightarrow x_0]\overline{x}.(\nu\,(x_0)))$. It is possible to prove that $(\boldsymbol{\nu} x_k)\,\nu\,([\delta\,(x_k) \leftrightarrow x_0]\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})) \equiv \nu\,([x_j \leftrightarrow x_0]\{\![\nu.\, q]\!\})$, for $x_j \notin \mathrm{fn}(\{\![\nu.\, q]\!\})$.

Rule (Open). By hypothesis, $\{\![p]\!\} \xrightarrow{\overline{x}\,x_0} \{\![q]\!\}$. Suppose $\delta\,(x_i) \notin \mathrm{fn}(\{\![p]\!\})$. Then, $(\boldsymbol{\nu} x_i)\,\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p]\!\}) \xrightarrow{\alpha} (\boldsymbol{\nu} x_i)\,\nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![q]\!\})$, with $\alpha = \nu\,([x_i \leftrightarrow x_0]\overline{x}\,)(\nu\,(\delta\,(x_i)))$, noting that $\nu\,([x_i \leftrightarrow x_0]\overline{x}) \neq x_i$ and$x \neq x_0$.

Proof of Lemma 3. The proof is by structural induction on the rules $\Rightarrow$. We only consider the most interesting cases.

As remarked, by applying axioms E_{pr} and E', every term p can be reduced into a normal form $p'(l_{s_1} \dots, l_{s_n})$, with ρ and δ not occurring in context $p'(\_, \_, \dots)$.

Rule (Rho) (similarly for (Rho′)). Let p, ρp, q, and ρq be in normal form. It is easy to see that p and ρp only differ on a finite number of name permutations applied to some s_i in their labels l_{s_i} and analogously for q and ρq. Then, $\mathcal{S}(\rho q) \subseteq \mathcal{S}(\rho p)$, by the hypothesis $\mathcal{S}(q) \subseteq \mathcal{S}(p)$.

Rule (Delta) (similarly for (Delta′)). Agents p and $\delta.\, p$ differ on a finite number of name substitutions. Then, the proof proceeds as in the previous case.

Rule (Static) (similarly for (Static′)). If $s \xrightarrow{\alpha} r$, the subagents of r are the same as s or less, except possibly for permutations and noninjective substitutions (the latter due to rule (π-Inp)).

Proof of Theorem 8. By construction, f, h, and $P_\Delta(h)$ are morphisms in $\mathbf{Alg}(\Sigma)$. Then, we have to prove that g is a morphism, i.e., $g(op^B(p_1, \dots, p_n)) = op^{P_\Delta(B)}(g(p_1), \dots, g(p_n))$.

Since h is surjective, there exist $t_1, \ldots, t_n$ such that $h(t_i) = p_i$, for $i = 1, \ldots, n$. Then,

$$\begin{aligned}
g(op^B(p_1, \ldots, p_n) &= g(op^B(h(t_1), \ldots, h(t_n)) \\
&= g(h(op^A(t_1, \ldots, t_n))) && (h \text{ morphism}) \\
&= P_\Delta(h)(f(op^A(t_1, \ldots, t_n))) && (\,h; g = f; P_L(h)\,) \\
&= op^{P_\Delta(B)}(P_\Delta(h)(f(t_1), \ldots, P_\Delta(h)(f(t_n)))) && (f; P_\Delta(h) \text{ morphism}) \\
&= op^{P_\Delta(B)}(g(h(t_1)), \ldots, g(h(t_n))) && (\,h; g = f; P_L(h)\,) \\
&= op^{P_\Delta(B)}(g(p_1)), \ldots, g(p_n)))
\end{aligned}$$

Proof of Theorem 9. The proof examines all the axioms E_{pr} and E'.

Ax. $\delta.\, \rho p = \rho_{+1}\delta.\, p$. There are two cases.

1. By rule (Delta), $\delta.\, \rho p \stackrel{\delta\,(\alpha)}{\Longrightarrow} \delta.\, p'$, with $\delta\,(\alpha) \neq \bar{x}, x$. Then, $\rho p \stackrel{\rho(\alpha')}{\Longrightarrow} \rho p''$ and $p \stackrel{\alpha'}{\Longrightarrow} p''$, with $\rho p'' = p'$, $\alpha = \rho(\alpha')$ and $\alpha' \neq \bar{x}', x'$, for any x'.
   On the other hand, by rule (Delta), $\delta.\, p \stackrel{\delta\,(\alpha')}{\Longrightarrow} \delta.\, p''$ with $\alpha' \neq \bar{x}', x'$ and, by rule (Rho), $\rho_{+1}\delta.\, p \stackrel{\rho_{+1}(\delta\,(\alpha')}{\Longrightarrow} \rho_{+1}\delta.\, p''$.
2. By rule (Delta′), suppose $\delta.\, \rho p \stackrel{\delta\,(\bar{x})}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p'$ (similarly, if $\alpha = \delta\,(x)$). Necessarily, $\rho p \stackrel{\rho(\bar{x}')}{\Longrightarrow} \rho_{+1}p''$ and $p \stackrel{\bar{x}'}{\Longrightarrow} p''$, with $\rho(\bar{x'}) = \bar{x}$ and $p' = \rho_{+1}p''$.
   On the other hand, by rule (Delta′), $\delta.\, p \stackrel{\delta\,(\bar{x}')}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p''$ and $\rho_{+1}\delta.\, p \stackrel{\rho_{+1}(\delta\,(\bar{x}'))}{\Longrightarrow} \rho_{+2}[x_0 \leftrightarrow x_1]\delta.\, p''$. Note that $\rho_{+2}[x_0 \leftrightarrow x_1]\delta.\, p'' = [x_0 \leftrightarrow x_1]\rho_{+2}\delta.\, p''$, as ρ_{+2} does not substitute either x_0 or x_1.

Ax. $\nu.\, (\delta.\, p)|q = p|\nu.\, q$. There are three cases.

1. By rule (Res), $\nu.\, (\delta.\, p)|q \stackrel{\nu\,(\alpha)}{\Longrightarrow} \nu.\, p'$. Necessarily, $(\delta.\, p)|q \stackrel{\alpha}{\Longrightarrow} p'$ and there are the following possible cases.
   (a) By rule (Par), suppose $\delta.\, p \stackrel{\alpha}{\Longrightarrow} p''$ (similarly, otherwise) and $p' = p''|q$. Then, $p \stackrel{\nu\,(\alpha)}{\Longrightarrow} p'''$, with $p'' = \delta.\, p'''$.
   On the other hand, by rule (Par), $p|\nu.\, q \stackrel{\nu\,(\alpha)}{\Longrightarrow} p'''|\nu.\, q$.

   (b) By rule (Com), $\delta.\, p \stackrel{\bar{x}\,y}{\Longrightarrow} p''$, $q \stackrel{xy}{\Longrightarrow} q''$, with $\alpha = \tau$ and $p' = p''|q''$. By rule (Delta), $p \stackrel{\nu\,(\bar{x}\,y)}{\Longrightarrow} p'''$, with $x_0 \notin \mathrm{n}(\bar{x}\,y)$ and $p'' = \delta.\, p'''$.
   On the other hand, by rule (Res), $\nu.\, q \stackrel{\nu\,(xy)}{\Longrightarrow} \nu.\, q''$ and, by rule (Com), $p|\nu.\, q \stackrel{\tau}{\Longrightarrow} p'''|\nu.\, q''$.

   (c) By rule (Close), $\alpha = \tau$. Suppose $\delta.\, p \stackrel{\bar{x}}{\Longrightarrow} p''$, $q \stackrel{x}{\Longrightarrow} q''$, and $p' = \nu.\, p''|q''$ (similar proof, otherwise). Then, by rule (Delta′), $p \stackrel{\nu\,(\bar{x})}{\Longrightarrow} p'''$, with $p'' = [x_0 \leftrightarrow x_1]\delta.\, p'''$.
   On the other hand, $\nu.\, q \stackrel{\nu\,(x)}{\Longrightarrow} \nu.\, q''$ and $p|\nu.\, q \stackrel{\tau}{\Longrightarrow} \nu.\, p'''|\nu.\, q''$, by rule (Close). Note that by axioms E_{pr} $\nu.\, \nu.\, [x_0 \leftrightarrow x_1](\delta.\, p''')|q'' = \nu.\, p'''|\nu.\, q''$.

2. By rule (RES'), $\nu.(\delta.p)|q \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$. By rule (PAR'), suppose $\delta.p \stackrel{\alpha}{\Longrightarrow} p''$, with $p' = p''|\delta.q$. By rule (DELTA'), $p \stackrel{\nu(\alpha)}{\Longrightarrow} p'''$ and $p'' = [x_0 \leftrightarrow x_1]\delta.p'''$.
   On the other hand, by rule (PAR'), $p|\nu.q \stackrel{\nu(\alpha)}{\Longrightarrow} p'''|\delta.\nu.q$. By axioms E_{pr}, $\nu.[x_0 \leftrightarrow x_1]([x_0 \leftrightarrow x_1]\delta.p''')|\delta.q = p'''|\delta.\nu.q$.

3. By rule (OPEN), $\nu.(\delta.p)|q \stackrel{\nu(\overline{x})}{\Longrightarrow} p'$. Necessarily, $\delta.p|q \stackrel{\bar{x}x_0}{\Longrightarrow} p'$ and $x \neq x_0$. By rule (PAR), suppose $q \stackrel{\bar{x}x_0}{\Longrightarrow} q'$ and $p' = \delta.p|q'$.
   On the other hand, by rule (OPEN), $\nu.q \stackrel{\nu(\bar{x})}{\Longrightarrow} q'$ and, by rule (PAR'), $p|\nu.q \stackrel{\nu(\bar{x})}{\Longrightarrow} (\delta.p)|q'$.

Ax. $\nu.\nu.[x_0 \leftrightarrow x_1]p = \nu.\nu.p$. There are the following possible cases.

1. By rule (RES), $\nu.\nu.[x_0 \leftrightarrow x_1]p \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.p'$ and $\nu.[x_0 \leftrightarrow x_1]p \stackrel{\alpha}{\Longrightarrow} p'$, with $\alpha \neq \bar{x}, x$ and $x_0 \notin \mathrm{n}(\alpha)$. Necessarily, rule (RES) has been applied to $\nu.[x_0 \leftrightarrow x_1]p$ and $[x_0 \leftrightarrow x_1]p \stackrel{\delta(\alpha)}{\Longrightarrow} p''$, with $p' = \nu.p''$ and $x_0 \notin \mathrm{n}(\delta(\alpha))$. By rule (RHO), $p \stackrel{\delta(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1]p''$.
   On the other hand, by rule (RES), $\nu.p \stackrel{\alpha}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$ and, again by rule (RES), $\nu.\nu.p \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.\nu.[x_0 \leftrightarrow x_1]p''$.
2. By rule (OPEN), $\nu.\nu.[x_0 \leftrightarrow x_1]p \stackrel{\nu(\overline{x})}{\Longrightarrow} p'$ and $\nu.[x_0 \leftrightarrow x_1]p \stackrel{\overline{x}x_0}{\Longrightarrow} p'$, with $x \neq x_0$. Necessarily, by rule (RES), $[x_0 \leftrightarrow x_1]p \stackrel{\delta(\overline{x})x_1}{\Longrightarrow} p''$, with $p' = \nu.p''$ and $x_0 \neq \delta(\overline{x}), \delta(x_0)$. By rule (RHO), $p \stackrel{\delta(\overline{x})x_0}{\Longrightarrow} [x_0 \leftrightarrow x_1]p''$.
   On the other hand, by rule (OPEN), $\nu.p \stackrel{\overline{x}}{\Longrightarrow} [x_0 \leftrightarrow x_1]p''$ and, by rule (RES'), $\nu.\nu.p \stackrel{\nu(\overline{x})}{\Longrightarrow} \nu.p''$.
3. By rule (RES'), $\nu.\nu.[x_0 \leftrightarrow x_1]p \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$ and $\nu.[x_0 \leftrightarrow x_1]p \stackrel{\alpha}{\Longrightarrow} p'$ with $\alpha = \bar{x}, x$ and $x_0 \notin \mathrm{n}(\alpha)$. By rule (RES'), $[x_0 \leftrightarrow x_1]p \stackrel{\delta(\alpha)}{\Longrightarrow} p''$, with $p' = \nu.[x_0 \mapsto x_1]p''$ and $x_0 \notin \mathrm{n}(\delta(\alpha))$. By rule (RHO'), $p \stackrel{\delta(\alpha)}{\Longrightarrow} [\delta(x_0) \leftrightarrow \delta(x_1)]p''$.
   On the other hand, by rule (RES'), $\nu.p \stackrel{\alpha}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1][\delta(x_0) \leftrightarrow \delta(x_1)]p''$ and, again by rule (RES'), $\nu.\nu.p \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]\nu.[x_0 \leftrightarrow x_1][\delta(x_0) \leftrightarrow \delta(x_1)]p''$.
4. By rule (RES'), $\nu.\nu.[x_0 \leftrightarrow x_1]p \stackrel{\nu(\alpha)}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$ and $\nu.[x_0 \leftrightarrow x_1]p \stackrel{\alpha}{\Longrightarrow} p'$ with $\alpha = \bar{x}, x$ and $x_0 \notin \mathrm{n}(\alpha)$. By rule (OPEN), $[x_0 \leftrightarrow x_1]p \stackrel{\delta(\overline{x})x_0}{\Longrightarrow} p'$, with $\alpha = \overline{x}$. By rule (RHO), $p \stackrel{\delta(\overline{x})x_1}{\Longrightarrow} [x_0 \leftrightarrow x_1]p'$.
   On the other hand, by rule (RES), $\nu.p \stackrel{\overline{x}x_0}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$ and, by rule (OPEN), $\nu.\nu.p \stackrel{\nu(\overline{x})}{\Longrightarrow} \nu.[x_0 \leftrightarrow x_1]p'$.

Ax. $\rho(p|q) = \rho p \,|\, \rho q$. The most interesting case is the following.

By rule (Rho$'$), $\rho(p|q) \stackrel{\rho(\overline{x})}{\Longrightarrow} \rho_{+1}p'$ and $p|q \stackrel{\overline{x}}{\Longrightarrow} p'$. Necessarily, by rule (Par$'$), $p \stackrel{\overline{x}}{\Longrightarrow} p''$ and $p' = p''|\delta.\, q$.

On the other hand, by rule (Rho$'$), $\rho p \stackrel{\rho(\overline{x})}{\Longrightarrow} \rho_{+1}p''$, and by rule (Par$'$) $\rho p \,|\, \rho q \stackrel{\rho(\overline{x})}{\Longrightarrow} (\rho_{+1}p'') \,|\, \delta.\, \rho q$.

Ax. $\rho\nu.\, p = \nu.\, \rho_{+1}p$. The most interesting case is as follows.

By rule (Rho$'$), $\rho\nu.\, p \stackrel{\rho(\alpha)}{\Longrightarrow} \rho_{+1}p'$ and $\nu.\, p \stackrel{\alpha}{\Longrightarrow} p'$ with $\alpha = \overline{x}, x$. By rule (Res$'$), $p \stackrel{\delta\,(\alpha)}{\Longrightarrow} p''$ and $p' = \nu.\, [x_0 \leftrightarrow x_1]p''$.

On the other hand, by rule (Rho$'$), $\rho_{+1}p \stackrel{\rho_{+1}(\delta\,(\alpha))}{\Longrightarrow} \rho_{+2}p''$, with $x_0 \notin \rho_{+1}(\delta\,(\alpha))$. Then, by rule (Res$'$), $\nu.\, \rho_{+1}p \stackrel{\rho(\alpha)}{\Longrightarrow} \nu.\, [x_0 \leftrightarrow x_1]\rho_{+2}p''$ and, by axioms E_{pr}, $\rho_{+1}\nu.\, [x_0 \leftrightarrow x_1]p'' = \nu.\, [x_0 \leftrightarrow x_1]\rho_{+2}p''$.

Ax. $\delta.\, p \,|\, q = (\delta.\, p) \,|\, \delta.\, q$. The most interesting case is the following.

By rule (Delta$'$), $\delta.\, p|q \stackrel{\delta\,(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p'$ and $p|q \stackrel{\alpha}{\Longrightarrow} p'$, with $\alpha = \overline{x}, x$. By rule (Par$'$), suppose $p \stackrel{\alpha}{\Longrightarrow} p''$ and $p' = p''|\delta.\, q$.

On the other hand, by rule (Delta$'$), $\delta.\, p \stackrel{\delta\,(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p''$ and, by rule (Par$'$), $(\delta.\, p) \,|\, \delta.\, q \stackrel{\delta\,(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1](\delta.\, p'') \,|\, \delta.\, \delta.\, q$.

Ax. $\delta.\, \nu.\, p = \nu.\, [x_0 \leftrightarrow x_1]\delta.\, p$.

By rule (Delta$'$), $\delta.\, \nu.\, p \stackrel{\delta\,(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p'$ and $\nu.\, p \stackrel{\alpha}{\Longrightarrow} p'$ with $\alpha = \overline{x}, x$. Necessarily, by (Res$'$), $p \stackrel{\delta\,(\alpha)}{\Longrightarrow} p''$, with $p' = \nu.\, [x_0 \leftrightarrow x_1]p''$.

On the other hand, by rule (Delta$'$), $\delta.\, p \stackrel{\delta\,(\delta\,(\alpha))}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, p''$. Then, by rule (Rho$'$), $[x_0 \leftrightarrow x_1]\delta.\, p \stackrel{\delta\,(\delta\,(\alpha))}{\Longrightarrow} [\delta\,(x_0) \leftrightarrow \delta\,(x_1)][x_0 \leftrightarrow x_1]\delta.\, p''$. Finally, by rule (Res$'$), $\nu.\, [x_0 \leftrightarrow x_1]\delta.\, p \stackrel{\delta\,(\alpha)}{\Longrightarrow} \nu.\, [x_0 \leftrightarrow x_1][\delta\,(x_0) \leftrightarrow \delta\,(x_1)][x_0 \leftrightarrow x_1]\delta.\, p''$.
By axioms E_{pr}, $[x_0 \leftrightarrow x_1]\delta.\, \nu.\, [x_0 \leftrightarrow x_1]p'' = \nu.\, [x_0 \leftrightarrow x_1][\delta\,(x_0) \leftrightarrow \delta\,(x_1)][x_0 \leftrightarrow x_1]\delta.\, p''$.

Ax. $\rho\, l_s = l_{\rho(s)}$.

By rule (Rho$'$), $\rho l_s \stackrel{\rho(\alpha)}{\Longrightarrow} \rho_{+1}q$ and $l_s \stackrel{\alpha}{\Longrightarrow} q$, with $\alpha = \overline{x}, x$. By rule (Static$'$), $s \stackrel{\{\![\alpha]\!\}}{\longrightarrow} \{\![q']\!\}$ and $q = [\delta\,(y) \leftrightarrow x_0]\delta.\, q'$, for $y \notin \mathrm{fn}(\{\![q']\!\})$.

On the other hand, by applying permutation ρ to transition $s \stackrel{\{\![\alpha]\!\}}{\longrightarrow} \{\![q']\!\}$, we obtain $\rho(s) \stackrel{\rho(\{\![\alpha]\!\})}{\longrightarrow} \rho(\{\![q']\!\})$. Then, by rule (Static$'$), $l_{\rho(s)} \stackrel{\rho(\alpha)}{\Longrightarrow} [\delta\,(\rho(y)) \leftrightarrow x_0]\delta.\, \rho q'$.

Ax. $\delta.\, l_s = l_{\delta\,(s)}$.

By rule (Delta$'$), $\delta.\, l_s \stackrel{\delta\,(\alpha)}{\Longrightarrow} [x_0 \leftrightarrow x_1]\delta.\, q$ and $l_s \stackrel{\alpha}{\Longrightarrow} q$, with $\alpha = \overline{x}, x$. Necessarily, by rule (Static$'$), $s \stackrel{\{\![\alpha]\!\}}{\longrightarrow} \{\![q']\!\}$ and $q = [\delta\,(y) \leftrightarrow x_0]\delta.\, q'$, for $y \notin \mathrm{fn}(\{\![q']\!\})$.
On the other hand, by applying function δ to each name in transition $s \stackrel{\{\![\alpha]\!\}}{\longrightarrow} \{\![q']\!\}$, $\delta\,(s) \stackrel{\delta\,(\{\![\alpha]\!\})}{\longrightarrow} \delta\,(\{\![q']\!\})$. Finally, by rule (Static$'$), $l_{\delta\,(s)} \stackrel{\delta\,(\alpha)}{\Longrightarrow} [\delta\,(\delta\,(y)) \leftrightarrow$

$x_0]\delta.\,\delta.\,q'$ and $[x_0 \leftrightarrow x_1]\delta.\,[\delta\,(y) \leftrightarrow x_0]\delta.\,q' = [\delta\,(\delta\,(y)) \leftrightarrow x_0]\delta.\,\delta.\,q'$, as $[x_0 \leftrightarrow x_1]$ is inactive on $\delta.\,\delta.\,q'$.

Proof of Theorem 10. We first prove that $p \sim q$ implies $[\![p]\!] \sim_g [\![q]\!]$.

We define a binary relation $\mathcal{R}$ as follows: $[\![p]\!]\,\mathcal{R}\,[\![q]\!] \stackrel{\text{def}}{\Longleftrightarrow} p \sim q$. We prove that $\mathcal{R}$ is a bisimulation. Suppose $[\![p]\!] \stackrel{\alpha}{\Longrightarrow} p'$. We only prove the most interesting case, i.e., with $\alpha = \bar{x}, x$, and we suppose $\alpha = \bar{x}$. By Lemma 2, $p \xrightarrow{\overline{x}(x_i)} \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p']\!\})$, $\forall \delta\,(x_i) \notin \mathrm{fn}(\{\![p']\!\})$, and, by definition of bisimilarity, $q \xrightarrow{\overline{x}(x_i)} q'$, with $q' \sim \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p']\!\})$. By Lemma 2, $[\![q]\!] \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(x_i) \leftrightarrow x_0]\delta.\,[\![q']\!]$. By Lemma 1, $q' \sim \nu\,([\delta\,(x_i) \leftrightarrow x_0]\{\![p']\!\})$ implies $[\delta\,(x_i) \leftrightarrow x_0]\delta\,(q') \sim \{\![p']\!\}$. Finally, by definition of $\mathcal{R}$, $[\delta\,(x_i) \leftrightarrow x_0]\delta.\,[\![q']\!]\,\mathcal{R}\,p'$, as required.

We now prove that $[\![p]\!] \sim_g [\![q]\!]$ implies $p \sim q$. We define a binary relation $\mathcal{R}$ as follows: $p\,\mathcal{R}\,q \stackrel{\text{def}}{\Longleftrightarrow} [\![p]\!] \sim_g [\![q]\!]$. We prove that $\mathcal{R}$ is a bisimulation. Suppose that $p \stackrel{\alpha}{\longrightarrow} p'$. We only prove the most interesting case, i.e., with $\alpha = \overline{x}(y), xy$, and we suppose $\alpha = \overline{x}(y)$. By Lemma 2, $[\![p]\!] \stackrel{\overline{x}}{\Longrightarrow} [\delta\,(y) \leftrightarrow x_0]\delta.\,[\![p']\!]$, and, by definition of bisimilarity, $[\![q]\!] \stackrel{\overline{x}}{\Longrightarrow} q'$, with $q' \sim_g [\delta\,(y) \leftrightarrow x_0]\delta.\,[\![p']\!]$. By Lemma 2, $q \xrightarrow{\overline{x}(y)} \nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q']\!\})$, assuming $\delta\,(y) \notin \mathrm{fn}(\{\![q']\!\})$. We have to prove that $\nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q']\!\})\,\mathcal{R}\,p'$. By Corollary 3, $\sim_g$ is a congruence and, thus, $q' \sim_g [\delta\,(y) \leftrightarrow x_0]\delta.\,[\![p']\!]$ implies $\nu.\,[\delta\,(y) \leftrightarrow x_0]q' \sim_g [\![p']\!]$. Finally, by definition of $\mathcal{R}$, $\nu\,([\delta\,(y) \leftrightarrow x_0]\{\![q']\!\})\,\mathcal{R}\,p'$, as required.

Proof of Corollary 4. First, we prove that if $p_1 \sim q_1$ and $p_2 \sim q_2$, then $p_1|p_2 \sim q_1|q_2$. By Theorem 10, $[\![p_1]\!] \sim_g [\![q_1]\!]$ and $[\![p_2]\!] \sim_g [\![q_2]\!]$. Since $\sim_g$ is a congruence, $[\![p_1]\!]|[\![p_2]\!] \sim_g [\![q_1]\!]|[\![q_2]\!]$ and, thus, $[\![p_1|p_2]\!] \sim_g [\![q_1|q_2]\!]$. Finally, by Theorem 10, $p_1|p_2 \sim q_1|q_2$.

Now, we prove that if $p \sim q$ then $(\boldsymbol{\nu}x)\,p \sim (\boldsymbol{\nu}x)\,q$. By Theorem 10, $[\![p]\!] \sim_g [\![q]\!]$. Since $\sim_g$ is a congruence, $\nu.\,[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![p]\!] \sim_g \nu.\,[\delta\,(x) \leftrightarrow x_0]\delta.\,[\![q]\!]$ and, by definition of $[\![]\!]$, $[\![(\boldsymbol{\nu}x)\,p]\!] \sim_g [\![(\boldsymbol{\nu}x)\,q]\!]$. Finally, by Theorem 10, $(\boldsymbol{\nu}x)\,p \sim (\boldsymbol{\nu}x)\,q$.