BRICS

Basic Research in Computer Science

New-HOPLA

A Higher-Order Process Language with Name Generation

Glynn Winskel
Francesco Zappa Nardelli

BRICS Report Series RS-04-21

ISSN 0909-0878 October 2004

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/04/21/`

new-HOPLA
a higher-order process language with name generation

Glynn Winskel
Computer Laboratory
University of Cambridge, UK
Glynn.Winskel@cl.cam.ac.uk

Francesco Zappa Nardelli
INRIA Rocquencourt
France
Francesco.Zappa_Nardelli@inria.fr

October 2004

Abstract

This paper introduces new-HOPLA, a concise but powerful language for higher-order nondeterministic processes with name generation. Its origins as a metalanguage for domain theory are sketched but for the most part the paper concentrates on its operational semantics. The language is typed, the type of a process describing the shape of the computation paths it can perform. Its transition semantics, bisimulation, congruence properties and expressive power are explored. Encodings are given of well-known process algebras, including π-calculus, Higher-Order π-calculus and Mobile Ambients.

1 The origins of new-HOPLA

This work is part of a general programme (reported in [8]), to develop a domain theory which scales up to the intricate languages, models and reasoning techniques used in distributed computation. This ambition led to a concentration on path based models, and initially on presheaf models because they can even encompass causal dependency models like event structures; so 'domains' is being understood more broadly than usual, to include presheaf categories.

The general methodology has been to develop domain theories with a rich enough life of their own to suggest powerful metalanguages. The point to emphasise is that in this way informative domain theories can have a pro-active role; they can yield new metalanguages, by their nature very expressive, accompanied by novel ways to deconstruct existing notions into more primitive ones, as well as new analysis techniques. A feature of presheaf models has been very useful: in key cases there is often a strong correspondence between elements of the presheaf denotation and derivations in an operational semantics. In the cases of HOPLA and new-HOPLA the presheaf models have led not only the core operations of the language, and a suitable syntax, but also to their operational semantics.

This paper reports on new-HOPLA, a compact but expressive language for higher-order nondeterministic processes with name generation. It extends the language HOPLA of Nygaard and Winskel [7] with name generation, and like its predecessor has its origins in a domain theory for concurrency. Specifically it arose out of the metalanguage implicitly being used in giving a presheaf semantics to the π-calculus [2]. But a sketch of its mathematical origins and denotational semantics does not require that heavy an investment, and can be based on path sets rather than presheaves.[1]

The key features of new-HOPLA hinge on its types and these can be understood independently of their origin as objects, and constructions on objects, in a category of domains—to be sketched shortly within the simple domain theory of path sets. A type $\mathbb{P}$ specifies the computations possible with respect to a given current set of names; if a process has type $\mathbb{P}$, then any computation path it performs with the current set of names s will be an element of $\mathbb{P}(s)$.

A central type constructor is that of prefix type $!\mathbb{P}$; at a current set of names s, a process of this type $!\mathbb{P}$, if it is to do anything, is constrained to first doing a prototypical action ! before resuming as a process of type $\mathbb{P}$. (Actions within sum or tensor types will come to be tagged by injections and so of a less anonymous character.)

In the category of domains, domains can be tensored together, a special case of which gives us types of the form $\mathbb{N} \otimes \mathbb{P}$, a kind of dynamic sum which at current names s comprises paths of $\mathbb{P}(s)$ tagged by a current name which serves as an injection function. There is also a more standard sum $\Sigma_{i \in I} \mathbb{P}_i$ of an indexed family of types $\mathbb{P}_i$ where $i \in I$; this time paths are tagged by indices from the fixed set I rather than the dynamic set of names.

The remaining type constructions are the formation of recursive types, and three forms of function space. One is a 'linear function space' $\mathbb{N} \to \mathbb{P}$, the type of processes which given a name return a process of type $\mathbb{P}$. Another is a 'continuous function space' $\mathbb{P} \to \mathbb{Q}$, the type of processes which given a process of type $\mathbb{P}$ return a process of type $\mathbb{Q}$. There is also a type $\delta\mathbb{P}$ associated directly with new-name generation. A process of type $\delta\mathbb{P}$ takes any new name (i.e. a name not in the current set of names) as input and returns a process of type $\mathbb{P}$. Name generation is represented by new name abstraction, to be thought of as picking a new name (any new name will do as well as any other), and resuming as a process in which that new name is current.

This summarises the rather economical core of new-HOPLA. Very little in the way of standard process algebra operations are built in—nothing beyond a prefix operation and nondeterministic sum. By being based on more fundamental primitives than usual, the language of new-HOPLA is remarkably expressive. As additional motivation we now turn to how these primitives arise from a mathematical model refining the intuitions we have just presented.

A domain theory If for the moment we ignore name generation, a suitable category of domains is that of **Lin**. Its objects, *path orders*, are preorders $\mathbb{P}$ consisting of computation paths with the order $p \leq p'$ expressing how a path p extends to a path p'. A path order $\mathbb{P}$ determines a domain $\widehat{\mathbb{P}}$, that of its *path sets*, left-closed sets w.r.t. $\leq_{\mathbb{P}}$, ordered by inclusion. (Such a domain is a prime-algebraic complete lattice, in which the complete primes are precisely those path sets generated by individual paths.) The arrows of **Lin**,

[1] Path sets arise by 'flattening' presheaves, which can be viewed as characteristic functions to truth values given in the category of sets, as sets of realisers, to simpler characteristic functions based on truth values $0 \leq 1$ [8].

linear maps, from $\mathbb{P}$ to $\mathbb{Q}$ are join-preserving functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$. The category **Lin** is monoidal-closed with a tensor given by the product $\mathbb{P} \times \mathbb{Q}$ of path orders and a corresponding function space by $\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}$—it is easy to see that join-preserving functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$ correspond to path sets of $\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}$. In fact **Lin** has enough structure to form a model of Girard's classical linear logic [4]. To exhibit its exponential ! we first define the category **Cts** to consist, like **Lin**, of path orders as objects but now with arrows the Scott-continuous functions between the domains of path sets. The inclusion functor $\mathbf{Lin} \hookrightarrow \mathbf{Cts}$ has a left adjoint $! : \mathbf{Cts} \to \mathbf{Lin}$ which takes a path order $\mathbb{P}$ to a path order consisting of finite subsets of $\mathbb{P}$ with order

$$P \leq_{!\mathbb{P}} P' \text{ iff } \forall p \in P\ \exists p' \in P'.\ p \leq_{\mathbb{P}} p'$$

—so $!\mathbb{P}$ can be thought of as consisting of compound paths associated with several runs.

The higher-order process language HOPLA is built around constructions in the category **Lin**. Types of HOPLA, which may be recursively defined, denote objects of **Lin**, path orders circumscribing the computation paths possible. As such all types support operations of nondeterministic sum and recursive definitions, both given by unions. Sum types are provided by coproducts, and products, of **Lin**, both given by the disjoint juxtaposition of path orders; they provide injection and projection operations. There is a type of functions from $\mathbb{P}$ to $\mathbb{Q}$ given by $(!\mathbb{P})^{\mathrm{op}} \times \mathbb{Q}$, the function space of **Cts**; this gives the operation of application and lambda abstraction. To this the adjunction yields a primitive prefix operation, a continuous map $\mathbb{P} \to !\mathbb{P}$, given by the unit at $\mathbb{P}$; it is accompanied by a destructor, a prefix-match operation, obtained from the adjunction's natural isomorphism. For further details, encodings of traditional process calculi in HOPLA and a full abstraction result, the reader is referred to [7, 9].

A domain theory for name generation We are interested in extending HOPLA to allow name generation. We get our inspiration from the domain theory. As usual a domain theory for name generation is obtained by moving to a category in which standard domains are indexed functorially by the current set of names. The category $\mathcal{I}$ consists of finite sets of names related by injective functions. The functor category $\mathbf{Lin}^{\mathcal{I}}$ has as objects functors $\mathbb{P} : \mathcal{I} \to \mathbf{Lin}$, so path orders $\mathbb{P}(s)$, indexed by finite sets of names s, standing for the computation paths possible with that current set of names; its arrows are natural transformations $\alpha = \langle \alpha_s \rangle_{s \in \mathcal{I}} : \mathbb{P} \to \mathbb{Q}$, with components in **Lin**. One important object in $\mathbf{Lin}^{\mathcal{I}}$ is the object of names $\mathbb{N}$ providing the current set of names, so $\mathbb{N}(s) = s$ regarded as a discrete order, at name set s. Types of new-HOPLA will denote objects of $\mathbf{Lin}^{\mathcal{I}}$.

The category has coproducts and products, both given by disjoint juxtaposition at each component. These provide a *sum type* $\Sigma_{i \in I}\mathbb{P}_i$ from a family of types $(\mathbb{P}_i)_{i \in I}$. It has *injections* producing a term $i{:}t$ of type $\Sigma_{i \in I}\mathbb{P}_i$ from a term t of type $\mathbb{P}_i$, for $i \in I$. *Projections* produce a term $\pi_i t$ of type $\mathbb{P}_i$ from a term t of the sum type.

There is a tensor got pointwise from the tensor of **Lin**. Given $\mathbb{P}$ and $\mathbb{Q}$ in $\mathbf{Lin}^{\mathcal{I}}$ we define $\mathbb{P} \otimes \mathbb{Q}$ in $\mathbf{Lin}^{\mathcal{I}}$ so that $(\mathbb{P} \otimes \mathbb{Q})(s) = \mathbb{P}(s) \times \mathbb{Q}(s)$ at $s \in \mathcal{I}$. We will only use a special case of this construction to form tensor types $\mathbb{N} \otimes \mathbb{P}$, so $(\mathbb{N} \otimes \mathbb{P})(s) = s \times \mathbb{P}(s)$ at $s \in \mathcal{I}$. These are a form of 'dynamic sum', referred to earlier, in which the components and the corresponding injections grow with the availability of new names. There are term constructors producing a term $n \cdot t$ of type $\mathbb{N} \otimes \mathbb{P}$ from a term t of type $\mathbb{P}$ and a name n. There are projections $\pi_n t$ forming a term of type $\mathbb{P}$ from a term t of tensor type.

At any stage s, the current set of names, a new name can be generated and used in a term in place of a variable over names. This leads to the central idea of new-name abstractions of type $\delta\mathbb{P}$ where $\delta\mathbb{P}(s) = \mathbb{P}(s \dot{\cup} \{\star\})$ at name set s. As observed by Stark [13] the construction $\delta\mathbb{P}$ can be viewed as a space of functions from $\mathbb{N}$ to $\mathbb{P}$ but with the proviso that the input name is fresh. A new-name abstraction is written $new\alpha.t$ and has type $\delta\mathbb{P}$, where t is a term of type $\mathbb{P}$. New-name application is written $t[n]$, where t has type $\delta\mathbb{P}$, and requires that the name n is fresh w.r.t. the names of t.

The adjunction $\mathbf{Lin} \underset{\hookrightarrow}{\overset{!}{\underset{\perp}{\longleftarrow}}} \mathbf{Cts}$ induces an adjunction $\mathbf{Lin}^{\mathcal{I}} \underset{\hookrightarrow}{\overset{!}{\underset{\perp}{\longleftarrow}}} \mathbf{Cts}^{\mathcal{I}}$ where the left adjoint is got by extending the original functor $! : \mathbf{Cts} \to \mathbf{Lin}$ in a pointwise fashion. The unit of the adjunction provides a family of maps from $\mathbb{P}$ to $!\mathbb{P}$ in $\mathbf{Cts}^{\mathcal{I}}$. As with HOPLA, these yield a prefix operation $!t$ of type $!\mathbb{P}$ for a term t of type $\mathbb{P}$. A type of the form $!\mathbb{P}$ is called a prefix type; its computation paths at any current name set first involve performing a prototypical action, also called '!'.

To support higher-order processes we need function spaces $\mathbb{P} \multimap \mathbb{Q}$ such that

$$\mathbf{Lin}^{\mathcal{I}}(\mathbb{R}, \mathbb{P} \multimap \mathbb{Q}) \;\cong\; \mathbf{Lin}^{\mathcal{I}}(\mathbb{R} \otimes \mathbb{P}, \mathbb{Q})$$

natural in $\mathbb{R}$ and $\mathbb{Q}$. Such function spaces do not exist in general—the difficulty is in getting a path order $\mathbb{P} \multimap \mathbb{Q}(s)$ at each name set s. However a function space does exist in the case where both $\mathbb{P}$ and $\mathbb{Q}$ satisfy certain 'type axioms' inherited through all the type operations and, in addition, $\mathbb{P}f$ preserves complete primes for each map $f : s \to s'$ in $\mathcal{I}$.[2] This suggests limiting the syntax of types to special function spaces $\mathbb{N} \multimap \mathbb{Q}$ and $!\mathbb{P} \multimap \mathbb{Q}$, the function space in $\mathbf{Cts}^{\mathcal{I}}$. The function spaces are associated with operations of application and lambda abstraction.

Related work and contribution The above domain theoretic constructions provide the basis of new-HOPLA. It resembles, and indeed has been inspired by, the metalanguages for domain theories with name generation used implicitly in earlier work [3, 13, 2], as well as the language of FreshML [11]. The language new-HOPLA is distinguished through the path-based domain theories to which it is fitted and, as we will see, in itself forming a process language with an operational semantics.

2 The language

Types The type of names is denoted by $\mathbb{N}$. The types of processes are defined by the grammar below.

$$\mathbb{P} \quad ::= \quad \mathbf{0} \;\big|\; \mathbb{N} \otimes \mathbb{P} \;\big|\; !\mathbb{P} \;\big|\; \delta\mathbb{P} \;\big|\; \mathbb{N} \to \mathbb{P} \;\big|\; \mathbb{P} \to \mathbb{Q} \;\big|\; \Sigma_{i \in I}\mathbb{P}_i \;\big|\; \mu_j P_1 \ldots P_k.(\mathbb{P}_1 \ldots \mathbb{P}_k) \;\big|\; P$$

[2] The type axioms, for $\mathbb{Q} : \mathcal{I} \to \mathbf{Lin}$, comprise: For every map $f : s \to s'$ in $\mathcal{I}$, the function $\mathbb{Q}f$ preserves finiteness and non-empty meets; preservation of non-empty meets ensures that whenever $y \subseteq \mathbb{Q}f(x)$ for some x, then there is a minimum $x_0 = \min(\mathbb{Q}f, y)$ for which $y \subseteq \mathbb{Q}f(x_0)$. For every pullback $h_1 : s_0 \to s_1, h_2 : s_0 \to s_2$ of $g_1 : s_1 \to s_3, g_2 : s_2 \to s_3$ in $\mathcal{I}$ and $x_1 \in \widehat{\mathbb{Q}s_1}$, if $\min(\mathbb{Q}g_2, \mathbb{Q}g_1(x_1))$ exists, then it is required that $\min(\mathbb{Q}h_1, x_1)$ exists and $\mathbb{Q}h_2(\min(\mathbb{Q}h_1, x_1)) = \min(\mathbb{Q}g_2, \mathbb{Q}g_1(x_1))$.

The sum type $\Sigma_{i \in I} \mathbb{P}_i$ where I is a finite set, is most often written $i_1{:}\mathbb{P} + \cdots + i_k{:}\mathbb{P}$. The symbol P is drawn from a set of type variables used in defining recursive types; closed type expressions are interpreted as path orders. The type $\mu_j P_1 \ldots P_k.(\mathbb{P}_1 \ldots \mathbb{P}_k)$ is interpreted as the j-component, for $1 \leq j \leq k$, of the 'least' solution to the defining equations $P_1 = \mathbb{P}_1, \ldots, P_k = \mathbb{P}_k$, where the expressions $\mathbb{P}_1 \ldots \mathbb{P}_k$ may contain the P_j's.

Terms and actions We assume a countably infinite set of *name constants*, ranged over by $a, b, \ldots$ and a countably infinite set of *name variables*, ranged over by $\alpha, \beta, \ldots$ Names, either constants or variables, are ranged over by $m, n, \ldots$. We assume an infinite, countable, set of *process variables*, ranged over by $x, y, \ldots$

Every type is associated with actions processes of that type may do. The *actions* are defined by the grammar below:

$$p, q, r \quad ::= \quad x \mid !p \mid n \cdot p \mid i{:}p \mid new\alpha.p \mid n \mapsto p \mid u \mapsto p \mid p[n] \ .$$

As we will see shortly, well-typed actions are constructed so that they involve exactly one prototypical action ! and exactly one 'resumption variable' x. Whenever a term performs the action, the variable of the action matches the resumption of the term: the typings of an action thus relates the type of a term with the type of its resumption. According to the transition rules a process of prefix type $!\mathbb{P}$ may do actions of the form $!p$, while a process of tensor or sum type may do actions of the form $n \cdot p$ or $i{:}p$ respectively. A process of type $\delta\mathbb{P}$ does actions of the form $new\alpha.p$ meaning that at the generation of a new name, a say, as input the action $p[a/\alpha]$ is performed. Actions of function type $n \mapsto p$ or $u \mapsto p$ express the dependency of the action on the input of a name n or process u respectively. The final clause is necessary in building up actions because we sometimes need to apply a resumption variable to a new name.

The *terms* are defined by the grammar below:

t, u, v	$::=$	$\mathbf{0}$	$!t$	inactive process and prototypical action
	$\mid$	$n \cdot t$	$\pi_n t$	tensor and projection
	$\mid$	$\lambda x.t$	tu	process abstraction and application
	$\mid$	$\lambda \alpha.t$	tn	name abstraction and application
	$\mid$	$new\alpha.t$	$t[n]$	new name abstraction and application
	$\mid$	$rec x.t$	x	recursive definition and process variables
	$\mid$	$i{:}t$	$\pi_i t$	injection and projection
	$\mid$	$\Sigma_{i \in I} t_i$	$\Sigma_{\alpha \in \mathbb{N}} t$	sum and sum over names
	$\mid$	$[t > p(x) \Rightarrow u]$		pattern matching

In new-HOPLA actions are used as patterns in terms $[t > p(x) \Rightarrow u]$ where we explicitly note the resumption variable x. If the term t can perform the action p the resumption of t is passed on to u via the variable x.

We assume an understanding of the *free name variables* (the binders of name variables are $\lambda\alpha.-$, $new\alpha.-$, and $\Sigma_{\alpha \in \mathbb{N}}-$) and of the *free process variables* (the binders of process variables are $\lambda x.-$, and $[t > p(x) \Rightarrow -]$) of a term. The *support* of a term, denoted $\mathsf{n}(t)$, is the set of the its free names, that is, the set of its name constants and of its free name variables.

We say that a name n is *fresh* for a term t if $n \notin \mathsf{n}(t)$.

2.1 Transition rules

The behaviour of terms is defined by a transition relation of the form

$$s \vdash t \xrightarrow{p\ (x)} t'$$

where s is a finite set of name constants such that $\mathsf{n}(t) \subseteq s$. The transition above should be read as 'with current names s the term t can perform the action p and resume as t''. We generally note the action's resumption variable in the transitions; this simplifies the transition rules in which the resumption variable must be explicitly manipulated.

So the transition relation is given at stages indexed by the set of current names s. The body of an abstraction over names $\lambda\alpha.t$ can only be instantiated with a name in s, and an abstraction over processes $\lambda x.t$ can only be instantiated with a process whose support is contained in s. As the transition relation is indexed by the current set of names, it is possible to generate new names at run-time. Indeed, the transition rule for new-name abstraction $new\alpha.t$ extends the set s of current names with a new name $a \notin s$; this name a is then passed to t via the variable α. The transition rules must respect the typings of actions and terms given in the next section. Formally:

Definition 2.1 (Transition relation) *For closed terms t such that $s \vdash t : \mathbb{P}$ and path patterns such that $s; ; x{:}\mathbb{Q} \Vdash p : \mathbb{P}$ the rules reported in Table 1 define a relation $\mathbb{P}; s \vdash t \xrightarrow{p\ (x)} u$, called the* transition relation.

To help familiarise the reader with the transition semantics, we present some derivations.

— *The operational semantics validates β-equivalence:*

$$\dfrac{\dfrac{\mathbb{Q}; s \vdash t[u/x] \xrightarrow{p\,(y)} t'}{\mathbb{P} \to \mathbb{Q}; s \vdash \lambda x.t \xrightarrow{u \mapsto p\,(y)} t'}}{\mathbb{Q}; s \vdash (\lambda x.t)u \xrightarrow{p\,(y)} t'} \qquad \dfrac{\dfrac{\mathbb{Q}; s \vdash t[a/\alpha] \xrightarrow{p\,(y)} t'}{\mathbb{N} \to \mathbb{Q}; s \vdash \lambda \alpha.t \xrightarrow{a \mapsto p\,(y)} t'}}{\mathbb{Q}; s \vdash (\lambda \alpha.t)a \xrightarrow{p\,(y)} t'}$$

These derivations illustrate how β-equivalence is validated by the transition relation in the sense that a term $(\lambda x.t)u$ (resp. $(\lambda\alpha.t)a$) has the same transition capabilities as the term $t[u/x]$ (resp. $t[a/\alpha]$): for example, a term $(\lambda x.t)u$ performs an action p and resumes as t' iff the term $t[u/x]$ performs the same action p resuming as t'—the 'only if' part follows by the uniqueness of the derivations.

— *The action of the tensor and sum type:*

$$\dfrac{\dfrac{\mathbb{P}; s \vdash t \xrightarrow{p\,(y)} t'}{\mathbb{N} \otimes \mathbb{P}; s \vdash a \cdot t \xrightarrow{a \cdot p\,(y)} t'}}{\mathbb{P}; s \vdash \pi_a(a \cdot t) \xrightarrow{p\,(y)} t'} \qquad \dfrac{\dfrac{\mathbb{P}_i; s \vdash t \xrightarrow{p\,(y)} t'}{\Sigma_{i \in I}\mathbb{P}_i; s \vdash i{:}t \xrightarrow{i:p\,(y)} t'}}{\mathbb{P}_i; s \vdash \pi_i(i{:}t) \xrightarrow{p\,(y)} t'}$$

By uniqueness of the derivations, a term t has the same transition capabilites as the terms $\pi_a(a \cdot t)$ and $\pi_i(i{:}t)$.

$$\frac{}{!\mathbb{P}; s \vdash !t \xrightarrow{!x\ (x)} t} \qquad \frac{\mathbb{P}; s \vdash t \xrightarrow{p} t' \quad a \in s}{\mathbb{N} \otimes \mathbb{P}; s \vdash a \cdot t \xrightarrow{a \cdot p} t'} \qquad \frac{\mathbb{N} \otimes \mathbb{P}; s \vdash t \xrightarrow{a \cdot p} t'}{\mathbb{P}; s \vdash \pi_a t \xrightarrow{p} t'}$$

$$\frac{\mathbb{P}; s \vdash t_i \xrightarrow{p\ (x)} t'}{\mathbb{P}; s \vdash \Sigma_{i \in I} t_i \xrightarrow{p\ (x)} t'} \qquad \frac{\mathbb{P}; s \vdash t[a/\alpha] \xrightarrow{p\ (x)} u \quad a \in s}{\mathbb{P}; s \vdash \Sigma_{\alpha \in \mathbb{N}} t \xrightarrow{p\ (x)} u} \qquad \frac{\mathbb{P}; s \vdash t[rec y.t/y] \xrightarrow{p\ (x)} u}{\mathbb{P}; s \vdash rec y.t \xrightarrow{p\ (x)} u}$$

$$\frac{\mathbb{P}_i; s \vdash t \xrightarrow{p\ (x)} t'}{\Sigma_{i \in I} \mathbb{P}_i; s \vdash i{:}t \xrightarrow{i:p\ (x)} t'} \qquad \frac{\Sigma_{i \in I} \mathbb{P}_i; s \vdash t \xrightarrow{i:p\ (x)} t'}{\mathbb{P}_i; s \vdash \pi_i t \xrightarrow{p\ (x)} t'} \qquad \frac{\mathbb{Q}; s \vdash t[u/x] \xrightarrow{p\ (x)} v \quad s \vdash u : \mathbb{P}}{\mathbb{P} \to \mathbb{Q}; s \vdash \lambda x.t \xrightarrow{u \mapsto p\ (x)} v}$$

$$\frac{\mathbb{P} \to \mathbb{Q}; s \vdash t \xrightarrow{u \mapsto p\ (x)} v}{\mathbb{Q}; s \vdash tu \xrightarrow{p\ (x)} v} \qquad \frac{\mathbb{P}; s \vdash t[a/\alpha] \xrightarrow{p\ (x)} v \quad a \in s}{\mathbb{N} \to \mathbb{P}; s \vdash \lambda \alpha.t \xrightarrow{a \mapsto p\ (x)} v} \qquad \frac{\mathbb{N} \to \mathbb{P}; s \vdash t \xrightarrow{a \mapsto p\ (x)} v}{\mathbb{P}; s \vdash ta \xrightarrow{p\ (x)} v}$$

$$\frac{\mathbb{P}; s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha]\ (x)} u[a/\alpha]}{\delta\mathbb{P}; s \vdash new\alpha.t \xrightarrow{new\alpha.p[x'[\alpha]/x]\ (x')} new\alpha.u} \qquad \frac{\delta\mathbb{P}; s \vdash t \xrightarrow{new\alpha.p[x'[\alpha]/x]\ (x')} u}{\mathbb{P}; s \mathbin{\dot\cup} \{a\} \vdash t[a] \xrightarrow{p[a/\alpha]\ (x)} u[a]}$$

$$\frac{\mathbb{P}; s \vdash t \xrightarrow{p\ (x)} t' \quad \mathbb{Q}; s \vdash u[t'/x] \xrightarrow{q\ (x')} v}{\mathbb{Q}; s \vdash [t > p(x) \Rightarrow u] \xrightarrow{q\ (x')} v}$$

In the rule for new name abstraction, the conditions $a \notin \mathsf{n}(p)$ and $a \notin \mathsf{n}(u)$ must hold.

Table 1: new-HOPLA: transition rules

— *New name abstraction and β-equivalence:*

$$\frac{\dfrac{\mathbb{Q}; s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha]\,(y)} t'[a/\alpha]}{\delta\mathbb{Q}; s \vdash new\alpha.t \xrightarrow{new\alpha.\alpha \mapsto p[y'[\alpha]/y]\,(y')} new\alpha.t'}}{\mathbb{Q}; s \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] \xrightarrow{p[a/\alpha]\,(y)} (new\alpha.t')[a]}$$

Again by uniqueness of the derivation, $(new\alpha.t)[a]$ and $t[a/\alpha]$ have the same transition capabilities for a fresh name a.

— *Matching the prefix action:*

$$\frac{!\mathbb{P}; s \vdash !t \xrightarrow{!x\,(x)} t \qquad \mathbb{Q}; s \vdash u[t/x] \xrightarrow{p\,(y)} u'}{\mathbb{Q}; s \vdash [!t > !x(x) \Rightarrow u] \xrightarrow{p\,(y)} u'}$$

The derivation above illustrates the simplest case of pattern matching. The term being tested $!t$ emits the prototypical action ! and continues as t. Then the computation path

$!x$ is matched against the path pattern $!x$; matching is successful and the continuation t is bound to x in u, which then executes. Pattern matching allows for the testing of arbitrary actions, even if a little care is needed when new names are involved.

— *Matching and new name abstraction:*

$$\frac{\dfrac{!\mathbb{P}; s \mathbin{\dot\cup} \{a\} \vdash\ !t[a/\alpha] \xrightarrow{!x'\,(x')} t[a/\alpha]}{\delta !\mathbb{P}; s \vdash \mathit{new}\alpha.!t \xrightarrow{\mathit{new}\alpha.!x[\alpha]\,(x)} \mathit{new}\alpha.t} \qquad \mathbb{Q}; s \vdash u[\mathit{new}\alpha.t/x] \xrightarrow{p\,(y)} u'}{\mathbb{Q}; s \vdash [\mathit{new}\alpha.!t > \mathit{new}\alpha.!x[\alpha](x) \Rightarrow u] \xrightarrow{p\,(y)} u'}$$

The resumption of a term of type $\delta\mathbb{P}$ after a transition is always a new name abstraction, and the new name generated in testing a pattern (above it is a) is local to the test.

2.2 Typing judgements

Consider a term $t = t'[\alpha]$. As we have discussed in the previous section, the square brackets denote new-name application: any name instantiating α should be fresh for the term t'. Consider now the context $C[-] = \lambda\alpha.-$. In the term $C[t] = \lambda\alpha.(t'[\alpha])$, the variable α is abstracted via a lambda abstraction, and may be instantiated with any current name. In particular it may be instantiated with names that belong to the support of t', thus breaking the hypothesis that t' has been applied to a fresh name. The same problem arises with contexts of the form $C[-] = \Sigma_{\alpha\in\mathbb{N}}-$. Moreover, if the process variable x is free in t, a context like $C[-] = \lambda x.-$ might instantiate x with an arbitrary term u. As the name instantiating α might belong to the support of u, nothing ensures it is still fresh for the term $t[u/x]$.

The type system must sometimes ensure that name variables are instantiated by fresh names. To impose this restriction, the typing context contains not only typing assumptions about name and process variables, such as $\alpha{:}\mathbb{N}$ and $x{:}\mathbb{P}$, but also *freshness assumptions* (or *distinctions*) about them, written (α, β) or (α, x). Here the intended meaning of (α, β) is that, in any environment, the names instantiating the variables α and β must be *distinct*. A freshness assumption like (α, x), where x is a process variable, records that in any environment the name instantiating α must be fresh for the term instantiating x.

Using this auxiliary information, the type system assumes that it is safe to abstract a variable, using lambda abstraction or sum over names, only if no freshness assumptions have been made on it.

The type system of new-HOPLA terms can be specified using judgements of the form:

$$A; \Gamma; d \vdash t : \mathbb{P}$$

where

- $A \equiv \alpha_1{:}\mathbb{N}, \ldots, \alpha_k{:}\mathbb{N}$ is a collection of name variables;
- $\Gamma \equiv x_1{:}\mathbb{P}_1, \ldots, x_k{:}\mathbb{P}_k$ is a partial function from of process variables, together with their types;

$$\frac{}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash\, !x : !\mathbb{R}} \qquad \frac{A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash \alpha\cdot p:\mathbb{N}\otimes\mathbb{P}}\alpha\in A$$

$$\frac{\alpha{:}\mathbb{N},A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}}{A;\Gamma;(d\setminus\alpha);;x'{:}\delta\mathbb{R}\Vdash new\alpha.p[x'[\alpha]/x]:\delta\mathbb{P}} \qquad \frac{A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash \alpha\mapsto p:\mathbb{P}}\alpha\in A$$

$$\frac{A;\Gamma;d\vdash u:\mathbb{Q} \qquad A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash u\mapsto p:\mathbb{Q}\to\mathbb{P}} \qquad \frac{A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}_j \quad j\in I}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash (j{:}p):\Sigma_{i\in I}\mathbb{P}_i}$$

$$\frac{A;\Gamma;d;;x{:}\mathbb{R}\Vdash t:\mathbb{P}_j[\mu\vec{P}.\vec{\mathbb{P}}/\vec{P}]}{A;\Gamma;d;;x{:}\mathbb{R}\Vdash t:\mu_jP:\vec{\mathbb{P}}} \qquad \frac{A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}}{A';\Gamma';d';;x{:}\mathbb{R}\Vdash p:\mathbb{P}}\begin{array}{l}A\subseteq A'\\ \Gamma\subseteq\Gamma'\\ d\subseteq d'\end{array}$$

Table 2: new-HOPLA: typing rules for actions

- d is a set of pairs $(\alpha,x)\in A\times\Gamma$, and $(\alpha,\beta)\in A\times A$, keeping track of the *freshness assumptions*.

Notation: We write $d\backslash\alpha$ for the set of freshness assumptions obtained from d by deleting all pairs containing α. The order in which variables appear in a distinction is irrelevant; we will write $(\alpha,\beta)\in d$ as a shorthand for $(\alpha,\beta)\in d$ *or* $(\beta,\alpha)\in d$. When we write $\Gamma\cup\Gamma'$ we allow the environments to overlap; the variables need not be disjoint provided the environments are consistent.

Actions are typed along the same lines, even if type judgements explicitly report the resumption variable:

$$A;\Gamma;d;;x{:}\mathbb{R}\Vdash p:\mathbb{P}\ .$$

The meaning of the environment $A;\Gamma;d$ is exactly the same as above. The variable x is the resumption variable of the pattern p, and its type is $\mathbb{R}$.

The type system of new-HOPLA is reported in Table 2 and Table 3.

The rule responsible for generating freshness assumptions is the rule for new-name application. If the term t has been typed in the environment $A;\Gamma;d$ and α is a new name variable (that is, $\alpha\notin A$), then the term $t[\alpha]$ is well-typed under the hypothesis that any name instantiating the variable α is distinct from all the names in terms instantiating the variables that can appear in t. This is achieved adding the set of freshness assumptions $\{\alpha\}\times(\Gamma\cup A)$ to d (when convenient, as here, we will confuse an environment with its domain).

The rule for pattern matching also modifies the freshness assumptions. The operational rule of pattern matching substitutes a subterm of t, whose names are contained in A', for x. Accordingly, the typing rule initially checks that no name in A' belongs to the set of the variables supposed fresh for x. Our attention is then drawn to the term $u[t'/x]$, where t' is a subterm of t. A name variable $\alpha\in A$ supposed fresh from x when typing u, must now be supposed fresh from all the free variables of t'. This justifies the freshness assumptions $\{\{\alpha\}\times(A'\cup\Gamma')\mid(\alpha,x)\in d\}$.

$$\frac{}{A;\Gamma;d\vdash\emptyset:\mathbb{P}}\qquad\frac{}{A;x{:}\mathbb{P},\Gamma;d\vdash x:\mathbb{P}}\qquad\frac{A;\Gamma;d\vdash t:\mathbb{P}}{A';\Gamma';d'\vdash t:\mathbb{P}}\begin{array}{l}A\subseteq A'\\ \Gamma\subseteq\Gamma'\\ d\subseteq d'\end{array}$$

$$\frac{\alpha{:}\mathbb{N},A;\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash\Sigma_{\alpha\in\mathbb{N}}t:\mathbb{P}}\alpha\notin d\qquad\frac{\alpha{:}\mathbb{N},A;\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash\lambda\alpha.t:\mathbb{N}\to\mathbb{P}}\alpha\notin d$$

$$\frac{A;x{:}\mathbb{Q},\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash\lambda x.t:\mathbb{Q}\to\mathbb{P}}x\notin d\qquad\frac{A;x{:}\mathbb{P},\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash rec x.t:\mathbb{P}}x\notin d$$

$$\frac{\alpha{:}\mathbb{N},A;\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;(d\setminus\alpha)\vdash new\alpha.t:\delta\mathbb{P}}\qquad\frac{A;\Gamma;d\vdash t:\delta\mathbb{P}}{\alpha{:}\mathbb{N},A;\Gamma;d\cup(\{\alpha\}\times(\Gamma\cup A))\vdash t[\alpha]:\mathbb{P}}$$

$$\frac{A;\Gamma;d\vdash t:\mathbb{N}\to\mathbb{P}}{A;\Gamma;d\vdash t\alpha:\mathbb{P}}\alpha\in A\qquad\frac{A;\Gamma;d\vdash t:\mathbb{P}\to\mathbb{Q}\quad A;\Gamma;d\vdash u:\mathbb{P}}{A;\Gamma;d\vdash tu:\mathbb{Q}}$$

$$\frac{A;\Gamma;d\vdash t_i:\mathbb{P}\quad\forall i\in I}{A;\Gamma;d\vdash\Sigma_{i\in I}t_i:\mathbb{P}}\qquad\frac{A;\Gamma;d\vdash t:\mathbb{P}_i}{A;\Gamma;d\vdash i{:}t:\Sigma_{i\in I}\mathbb{P}_i}\qquad\frac{A;\Gamma;d\vdash t:\Sigma_{i\in I}\mathbb{P}_i}{A;\Gamma;d\vdash\pi_i t:\mathbb{P}_i}\qquad\frac{A;\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash !t:!\mathbb{P}}$$

$$\frac{A;\Gamma;d\vdash t:\mathbb{P}}{A;\Gamma;d\vdash\alpha\cdot t:\mathbb{N}\otimes\mathbb{P}}\alpha\in A\qquad\frac{A;\Gamma;d\vdash t:\mathbb{N}\otimes\mathbb{P}}{A;\Gamma;d\vdash\pi_\alpha t:\mathbb{P}}\alpha\in A\qquad\frac{A;\Gamma;d\vdash t:\mathbb{P}_j[\mu\vec{P}.\vec{\mathbb{P}}/\vec{P}]}{A;\Gamma;d\vdash t:\mu_j P:\vec{\mathbb{P}}}$$

$$\frac{A';\Gamma';d'\vdash t:\mathbb{P}\qquad A';\Gamma';d';;x{:}\mathbb{R}\Vdash p:\mathbb{P}\qquad A;x{:}\mathbb{R},\Gamma;d\vdash u:\mathbb{Q}}{A\cup A';\Gamma\cup\Gamma';\overline{d}\vdash[t>p(x)\Rightarrow u]:\mathbb{Q}}A'\cap\{\alpha\mid(\alpha,x)\in d\}=\emptyset$$

$$\text{where }\overline{d}=(d\setminus x)\cup d'\cup\{\{\alpha\}\times(A'\cup\Gamma')\mid(\alpha,x)\in d\}$$

Table 3: new-HOPLA: typing rules for processes

The rest of the type system follows along the lines of type systems for the simply typed λ-calculus.

The type system assumes that terms do not contain name constants. This is to avoid the complications in a type system coping with both name variables and constants at the same time. We write $s\vdash t:\mathbb{P}$ when there is a judgement $A;\emptyset;d\vdash\sigma t':\mathbb{P}$ and a substitution σ for A respecting the distinctions d such that t is $\sigma t'$. In particular, $s\vdash t:\mathbb{P}$ iff there is a canonical judgement $A;\emptyset;\{(\alpha,\beta)\mid\alpha\neq\beta\}\vdash t':\mathbb{P}$, in which the substitution σ is a bijection between name variables and names and t is $\sigma t'$. Similarly for patterns.

We can now prove that the operational rules are type correct.

Lemma 2.2 (Substitution Lemma)

1. *Suppose that* $A';\Gamma';d'\vdash t:\mathbb{Q}$ *and* $A;x{:}\mathbb{Q},\Gamma;d;;y{:}\mathbb{R}\Vdash p:\mathbb{P}$*, where* $\Gamma\cup\Gamma'$ *is consistent and* $A'\cap\{\alpha\mid(\alpha,x)\in d\}=\emptyset$*. Then,*

$$A\cup A';\Gamma\cup\Gamma';\overline{d};;y{:}\mathbb{R}\Vdash p[t/x]:\mathbb{P}$$

where $\overline{d} = (d \setminus x) \cup d' \cup \{\{\alpha\} \times (A' \cup \Gamma') \mid (\alpha, x) \in d\}$.

2. *Suppose that* $A'; \Gamma'; d' \vdash t : \mathbb{Q}$ *and* $A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{P}$*, where* $\Gamma \cup \Gamma'$ *is consistent and* $A' \cap \{\alpha \mid (\alpha, x) \in d\} = \emptyset$*. Then,*

$$A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{P}$$

where $\overline{d} = (d \setminus x) \cup d' \cup \{\{\alpha\} \times (A' \cup \Gamma') \mid (\alpha, x) \in d\}$.

Lemma 2.3 (Contraction Rules) *The rules below can be derived (also for patterns):*

$$\frac{\alpha{:}\mathbb{N}, \beta{:}\mathbb{N}, A; \Gamma; d \vdash p : \mathbb{P}}{\alpha{:}\mathbb{N}, A; \Gamma; d[\alpha/\beta] \vdash p[\alpha/\beta] : \mathbb{P}} (\alpha, \beta) \notin d \qquad \frac{A; y{:}\mathbb{Q}, z{:}\mathbb{Q}, \Gamma; d \vdash p : \mathbb{P}}{A; y{:}\mathbb{Q}, \Gamma; d[y/z] \vdash p[y/z] : \mathbb{P}}$$

Theorem 2.4 (Transitions preserve types) *If* $s \vdash t : \mathbb{P}$ *and* $s; ; x{:}\mathbb{R} \Vdash p : \mathbb{P}$ *and* $\mathbb{P}; s \vdash t \xrightarrow{p\ (x)} t'$*, then* $s \vdash t' : \mathbb{R}$.

Some simple results follow. They can be proved by routine inductions on derivations of transition.

Lemma 2.5 *If* $\mathsf{n}(t) \subseteq s$ *and* $\mathbb{P}; s \vdash t \xrightarrow{p} t'$*, then* $\mathsf{n}(t') \subseteq s$.

Lemma 2.6 *If* $\mathbb{P}; s \vdash t \xrightarrow{p} t'$*, then* $\mathsf{n}(t) \cup \mathsf{n}(p) \cup \mathsf{n}(t') \vdash t \xrightarrow{p} t'$.

Lemma 2.7 (Injective renaming) *If* $\mathbb{P}; s \vdash t \xrightarrow{p} t'$ *and* $f : s \to s'$ *is injective, then* $\mathbb{P}; s' \vdash ft \xrightarrow{fp} ft'$.

Lemma 2.8 (Converse of injective renaming) *Let* $f : s \to s'$ *injective. If* $\mathbb{P}; s' \vdash ft \xrightarrow{p'} u'$*, then there exists* p, u *and* $g : s'' \to \mathsf{n}(p', u') \setminus \mathrm{ran}(f)$ *bijective, with* $s'' \cap s = \emptyset$*, such that* $\mathbb{P}; s, s'' \vdash t \xrightarrow{p} u$ *and* $p' = (f+g)p$ *and* $u' = (f+g)u$.

Observe that if $s' \vdash ft \xrightarrow{p'} u'$ then ft can chose nondeterministically some names in $(s' \setminus s)$ before performing p' (because of the semantics of the *sum over names*). This motivates the need for s'' in the Lemma above. As a consequence, bisimulation techniques based on injective renaming cannot be applied to new-HOPLA.

3 Equivalences

After introducing some notations regarding relations, we explore the bisimulation equivalence that arises from the transition semantics.

A relation $\mathcal{R}$ between typing judgements is said to respect types if, whenever $\mathcal{R}$ relates $E_1 \vdash t_1 : \mathbb{P}_1$ and $E_2 \vdash t_2 : \mathbb{P}_2$, we have $E_1 = E_2$ and $\mathbb{P}_1 = \mathbb{P}_2$. We are mostly interested in relations between closed terms, and we write $s \vdash t\ \mathcal{R}\ u : \mathbb{P}$ to denote $(s \vdash t : \mathbb{P}, s \vdash q : \mathbb{P}) \in \mathcal{R}$.

Definition 3.1 (Bisimilarity) *A type-respecting relation on closed terms,* $\mathcal{R}$*, is a* bisimulation *if*

1. $s \vdash t \mathcal{R} u : \mathbb{P}$ *and* $\mathbb{P}; s' \vdash t \xrightarrow{p(x)} t'$ *for* $s' \supseteq s$ *imply that there exists a term* u' *such that* $\mathbb{P}; s' \vdash u \xrightarrow{p(x)} u'$ *and* $s' \vdash t' \mathcal{R} u' : \mathbb{R}$*;*

2. $s \vdash t \mathcal{R} u : \mathbb{P}$ *and* $\mathbb{P}; s' \vdash u \xrightarrow{p(x)} u'$ *for* $s' \supseteq s$ *imply that there exists a term* t' *such that* $\mathbb{P}; s' \vdash t \xrightarrow{p(x)} t'$ *and* $s' \vdash t' \mathcal{R} u' : \mathbb{R}$*;*

where $\mathbb{R}$ *is the type of the resumption variable* x *in* p*. Let* bisimilarity*, denoted* $\sim$*, be the largest bisimulation.*

We say that two closed terms t and q are bisimilar if $s \vdash t \sim q : \mathbb{P}$ for some s and $\mathbb{P}$.

In the definition of bisimulation, the universal quantification on sets of names s' is required, otherwise we would relate

$$\{a\} \vdash \lambda\alpha.[\alpha!\mathbf{0} > a!x \Rightarrow !\mathbf{0}] : \mathbb{N} \otimes\ !\mathbf{0} \qquad \text{and} \qquad \{a\} \vdash \lambda\alpha.!\mathbf{0} : \mathbb{N} \otimes\ !\mathbf{0}$$

while the two terms above behave differently in a world where a is not the only current name.

Using an extension of Howe's method [6] as adapted by Gordon and Pitts to a typed setting [5, 10], we show that bisimilarity is preserved by well typed contexts.

Theorem 3.2 *Bisimilarity* $\sim$ *is an equivalence relation and a congruence.*

Proposition 3.3 *For closed, well-formed, terms we have*

$$\begin{array}{cc}
s \vdash (\lambda x.t)u \sim t[u/x] : \mathbb{P} & s \vdash (\lambda\alpha.t)a \sim t[a/\alpha] : \mathbb{P} \\
s \vdash \lambda x.(tx) \sim t : \mathbb{P} \to \mathbb{Q} & s \vdash \lambda\alpha.(t\alpha) \sim t : \mathbb{N} \to \mathbb{P} \\
s \vdash \lambda x.(\Sigma_{i\in I} t_i) \sim \Sigma_{i\in I}(\lambda x.t_i) : \mathbb{P} \to \mathbb{Q} & s \vdash \lambda\alpha.(\Sigma_{i\in I} t_i) \sim \Sigma_{i\in I}(\lambda\alpha.t_i) : \mathbb{N} \to \mathbb{P} \\
s \vdash (\Sigma_{i\in I} t_i)u \sim \Sigma_{i\in I}(t_i u) : \mathbb{P} & s \vdash (\Sigma_{i\in I} t_i)a \sim \Sigma_{i\in I}(t_i a) : \mathbb{P}
\end{array}$$

$$\begin{array}{cc}
s \vdash \pi_\beta(\beta \cdot t) \sim t : \mathbb{P} & s \vdash \pi_\beta(\alpha \cdot t) \sim \mathbf{0} : \mathbb{P} \\
\multicolumn{2}{l}{s \vdash t \sim \Sigma_{\alpha\in\mathbb{N}} \alpha \cdot (\pi_\alpha t) : \mathbb{N} \otimes \mathbb{P}} \\
s \vdash \beta \cdot (\Sigma_{i\in I} t_i) \sim \Sigma_{i\in I} \beta \cdot t_i : \mathbb{P} & s \vdash \pi_\beta(\Sigma_{i\in I} t_i) \sim \Sigma_{i\in I} \pi_\beta t_i : \mathbb{P}
\end{array}$$

$$s \vdash [!u > !x \Rightarrow t] \sim t[u/x] : \mathbb{P} \qquad s \vdash [\Sigma_{i\in I} u_i > !x \Rightarrow t] \sim \Sigma_{i\in I}[u_i > !x \Rightarrow t] : \mathbb{P}$$

Proposition 3.4 *Bisimilarity validates* β*-reduction on new-name abstraction:*

$$s \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] \sim t[a/\alpha] : \mathbb{P} \ .$$

Corollary 3.5 *If* $s \vdash new\alpha.t \sim new\alpha.u : \delta\mathbb{P}$ *then* $s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \sim u[a/\alpha] : \mathbb{P}$ *for all* $a \notin \mathsf{n}(t, u)$*.*

Proposition 3.6 (β-equivalence on new names) *Let* t *be a term with* α *free, that is* $A, \alpha; \emptyset; A, d \vdash t : \mathbb{P}$*. Let* $\sigma : s \to A$ *be a bijection. Then*

$$s \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] \sim t[a/\alpha] : \mathbb{P} \ .$$

Corollary 3.7 *If* $s \vdash new\alpha.t \sim new\alpha.u : \delta\mathbb{P}$*, then* $s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \sim u[a/\alpha] : \mathbb{P}$*.*

Corollary 3.8 *Let* t *and* u *be terms such that* $A, \alpha{:}\mathbb{N}; \emptyset; A, d \vdash t : \mathbb{P}$*, let* $\sigma : s \to A$ *be a bijection, and let* a, a' *be names not in* s*. If* $s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \sim u[a/\alpha] : \mathbb{P}$ *then* $s \mathbin{\dot\cup} \{a'\} \vdash t[a'/\alpha] \sim u[a'/\alpha] : \mathbb{P}$*.*

Proof By Proposition 3.5 $s \vdash new\alpha.t \sim new\alpha.u : \delta\mathbb{P}$. The result follows by Corollary 3.7. $\square$

4 Examples

In this section, we illustrate how new-HOPLA can be used to give semantics to well-known process algebras. We define a 'fully abstract' encoding of π-calculus that preserves and reflects both the reduction relation and strong bisimilarity. We also report an encoding of Higher-Order π-calculus and of Mobile Ambients.

We introduce an useful product type $\mathbb{P} \,\&\, \mathbb{Q}$, which is not primitive in new-HOPLA. It is definable as $1{:}\mathbb{P} + 2{:}\mathbb{Q}$. The projections are given by $\mathit{fst}(t) = \pi_1(t)$ and $\mathit{snd}(t) = \pi_2(t)$, while pairing is defined as $(t, u) = 1{:}t + 2{:}u$. For actions $(p, -) = 1{:}p$, $(-, q) = 2{:}q$. It is then easy to verify that $s \vdash \mathit{fst}(t, u) \sim t : \mathbb{P}$, that $s \vdash \mathit{snd}(t, u) \sim u : \mathbb{Q}$, and that $s \vdash (\mathit{fst}(t, u), \mathit{snd}(t, u)) \sim (t, u) : \mathbb{P} \,\&\, \mathbb{Q}$, for all $s \supseteq \mathsf{n}(t) \cup \mathsf{n}(u)$.

4.1 π-calculus

We denote *name constants* with $a, b, \ldots$, and *name variables* with $\alpha, \beta, \ldots$; the letters $n, m, \ldots$ range over both name constants and name variables. The terms of the language are constructed according the following grammar:

$$P, Q \;::=\; \mathbf{0} \;\big|\; P \mid Q \;\big|\; (\boldsymbol{\nu}\alpha)P \;\big|\; \overline{n}m.P \;\big|\; n(\alpha).P \;.$$

The late labelled transition system (denoted $\xrightarrow{\alpha}_l$) and the definition of strong late bisimulation (denoted $\sim_l$) are standard [12], and for reference are reported in Appendix A.3.

We can specify a type $\mathbb{P}$ as

$$\mathbb{P} \;=\; \tau{:}!\mathbb{P} \;+\; \mathsf{out}{:}\mathbb{N} \otimes \mathbb{N} \otimes !\mathbb{P} \;+\; \mathsf{bout}{:}\mathbb{N} \otimes !(\delta\mathbb{P}) \;+\; \mathsf{inp}{:}\mathbb{N} \otimes !(\mathbb{N} \to \mathbb{P}) \;.$$

The terms of π-calculus can be expressed in new-HOPLA as the following terms of type $\mathbb{P}$:

$$[\![\mathbf{0}]\!] = \mathbf{0} \qquad [\![\overline{n}m.P]\!] = \mathsf{out}{:}n \cdot m \cdot ![\![P]\!] \qquad [\![n(\beta).P]\!] = \mathsf{inp}{:}n \cdot !(\lambda\beta.[\![P]\!])$$
$$[\![(\boldsymbol{\nu}\alpha)P]\!] = \mathit{Res}\,(\mathit{new}\alpha.[\![P]\!]) \qquad [\![P \mid Q]\!] = [\![P]\!] \;\;||\;\; [\![Q]\!]$$

Here, $\mathit{Res} : \delta\mathbb{P} \to \mathbb{P}$ and $|| : \mathbb{P}\&\mathbb{P} \to \mathbb{P}$ (we use infix notation for convenience) and are abbrevations for the following recursively defined processes:

$$\begin{aligned}
\mathit{Res}\ t \;\; = \;\; & [t > \mathit{new}\alpha.\tau{:}!(x[\alpha])) \Rightarrow \tau{:}!\mathit{Res}\,x] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}\Sigma_{\gamma\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{out}{:}\beta \cdot \gamma \cdot !(x[\alpha]) \Rightarrow \mathsf{out}{:}\beta \cdot \gamma \cdot !\mathit{Res}\,x] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{out}{:}\beta \cdot \alpha \cdot !(x[\alpha]) \Rightarrow \mathsf{bout}{:}\beta \cdot !x] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{bout}{:}\beta \cdot !(x[\alpha]) \Rightarrow \mathsf{bout}{:}\beta \cdot !\mathit{new}\gamma \cdot \mathit{Res}\,(\mathit{new}\eta.x[\eta][\gamma])] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{inp}{:}\beta \cdot !(x[\alpha]) \Rightarrow \mathsf{inp}{:}\beta \cdot !\lambda\gamma.\mathit{Res}\,(\mathit{new}\eta.x[\eta](\gamma))]
\end{aligned}$$

$$\begin{aligned}
t \;\; || \;\; u \;\; = \;\; & [t > \tau{:}!x \Rightarrow \tau{:}!(x \;\; || \;\; u)] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}\Sigma_{\gamma\in\mathbb{N}}[t > \mathsf{out}{:}(\beta \cdot \gamma \cdot !x) \Rightarrow [u > \mathsf{inp}{:}(\beta \cdot !y) \Rightarrow \tau{:}!(x \;\; || \;\; y\gamma)]] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{bout}{:}(\beta \cdot !x) \Rightarrow [u > \mathsf{inp}{:}(\beta \cdot !y) \Rightarrow \tau{:}!\mathit{Res}\,(\mathit{new}\eta.(x[\eta] \;\; || \;\; y\eta))]] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}\Sigma_{\gamma\in\mathbb{N}}[t > \mathsf{out}{:}\beta \cdot \gamma \cdot !x \Rightarrow \mathsf{out}{:}\beta \cdot \gamma \cdot !(x \;\; || \;\; u)] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{bout}{:}\beta \cdot !x \Rightarrow \mathsf{bout}{:}\beta \cdot !\mathit{new}\eta.(x[\eta] \;\; || \;\; u)] \\
+ \;\; & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{inp}{:}\beta \cdot !x \Rightarrow \mathsf{inp}{:}\beta \cdot !\lambda\eta.(x(\eta) \;\; || \;\; u)] \\
+ \;\; & \text{symmetric cases}
\end{aligned}$$

where η is chosen such that $\eta \notin \mathsf{n}(u)$. Informally, the restriction map $Res : \delta\mathbb{P} \to \mathbb{P}$ pushes restrictions inside processes as far as possible. The five summands corresponds to the five equations below:

$$(\boldsymbol{\nu}\alpha)\tau.P \sim_l \tau.(\boldsymbol{\nu}\alpha)P$$

$$(\boldsymbol{\nu}\alpha)\overline{m}n.P \sim_l \overline{m}n.(\boldsymbol{\nu}\alpha)P \;\text{ if } \alpha \neq m,n \qquad (\boldsymbol{\nu}\alpha)\overline{m}\alpha.P \sim_l \overline{m}(\alpha).P \;\text{ if } \alpha \neq m$$

$$(\boldsymbol{\nu}\alpha)\overline{m}(\beta).P \sim_l \overline{m}(\beta).(\boldsymbol{\nu}\alpha)P \;\text{ if } \alpha \neq m \qquad (\boldsymbol{\nu}\alpha)m\beta.P \sim_l m\beta.(\boldsymbol{\nu}\alpha)P \;\text{ if } \alpha \neq m$$

where $\overline{m}(\alpha)$ is an abbreviation to express bound-output, that is, $(\boldsymbol{\nu}\alpha)\overline{m}\alpha$. The map Res implicitly also ensures that $(\boldsymbol{\nu}\alpha)P \sim_l 0$ if none of the above cases applies. The parallel composition map $||$ captures the *(late) expansion law* of π-calculus. There is a strong correspondence between actions performed by a closed π-calculus process and the actions of its encoding.

Theorem 4.1 *Let P a closed π-calculus process. If $P \xrightarrow{\tau}_l P'$ is derivable in π-calculus, then $\mathsf{n}([\![P]\!]) \vdash [\![P]\!] \xrightarrow{\tau:!}\sim [\![P']\!]$. Conversely, if $\mathsf{n}([\![P]\!]) \vdash [\![P]\!] \xrightarrow{\tau:!} t$ in new-HOPLA, then $P \xrightarrow{\tau}_l P'$ for some P', and $\mathsf{n}(t) \vdash t \sim [\![P']\!] : \mathbb{P}$.*

The encoding also preserves and reflects late strong bisimulation.

Theorem 4.2 *Let P and Q be two closed π-calculus processes. If $P \sim_l Q$ then $\mathsf{n}(P) \cup \mathsf{n}(Q) \vdash [\![P]\!] \sim [\![Q]\!] : \mathbb{P}$. Conversely, if $\mathsf{n}([\![P]\!]) \cup \mathsf{n}([\![Q]\!]) \vdash [\![P]\!] \sim [\![Q]\!] : \mathbb{P}$, then $P \sim_l Q$.*

Early semantics Along the same lines, new-HOPLA can encode the early semantics of π-calculus. The type of the input action assigned to π-calculus terms captures the difference between the two semantics. In the late semantics a process performing an input action has type $\mathsf{inp}{:}\mathbb{N} \otimes !(\mathbb{N} \to \mathbb{P})$: the type of the continuation $(\mathbb{N} \to \mathbb{P})$ ensures that the continuation is actually an *abstraction* that will be instantiated with the received name when interaction takes place. In the early semantics, the type of a process performing an input action is changed into $\mathsf{inp}{:}\mathbb{N} \otimes \mathbb{N} \to !\mathbb{P}$. Performing an input action now involves picking up a name before executing the prototypical action, and in the continuation (whose type is $\mathbb{P}$) the formal variable has been instantiated with the received name. Details can be found in [15].

Polyadic π-calculus A natural and convenient extension to π-calculus is to admit processes that pass tuples of names: polyadicity is a good testing ground for the expressivity of our language.

We can specify a type for polyadic π-calculus processes as:

$$\begin{aligned}
\mathbb{P} &= \tau{:}!\mathbb{P} \;+\; \mathsf{out}{:}\mathbb{N} \otimes \mathbb{C} \;+\; \mathsf{inp}{:}\mathbb{N} \otimes !\mathbb{F} \\
\mathbb{C} &= 0{:}\mathbb{N} \otimes \mathbb{C} \;+\; 1{:}\delta\mathbb{C} \;+\; 2{:}!\mathbb{P} \\
\mathbb{F} &= 3{:}\mathbb{N} \to \mathbb{F} \;+\; 4{:}\mathbb{P}
\end{aligned}$$

Recursive types are used to encode tuples of (possibly new) names in concretions, and sequences of name abstractions in abstractions.

Just as with the π-calculus, it is possible to write a restriction map $Res : \delta\mathbb{P} \to \mathbb{P}$ that pushes restrictions inside processes as far as possible, and a parallel map that captures the expansion law. The resulting semantics coincides with the standard late semantics of polyadic π-calculus. Details can be found in [15].

4.2 Higher-Order π-calculus

The language we consider can be found in [12], with one main difference: rather than introducing a unit value, we allow processes in addition to abstractions to be communicated. For brevity, we gloss over typing issues. The syntax of terms and values is defined below.

$$P, Q \quad ::= \quad V \bullet W \;\big|\; n(x).P \;\big|\; \overline{n}(V).P \;\big|\; P \mid Q \;\big|\; x \;\big|\; (\boldsymbol{\nu}\alpha)P \;\big|\; \mathbf{0}$$

$$V, W \quad ::= \quad P \;\big|\; (x).P$$

The reduction semantics for the language is standard [12]; we only recall the axioms that define the reduction relation:

$$(x).P \bullet V \;\rightharpoonup\; P[V/x] \qquad\qquad \overline{n}(V).P \mid n(x).Q \;\rightharpoonup\; P \mid Q[V/x]\ .$$

Types for HOπ are given recursively by

$$\mathbb{P} = \tau{:}!\mathbb{P} + \mathsf{out}{:}\mathbb{N}\otimes!\mathbb{C} + \mathsf{inp}{:}\mathbb{N}\otimes!(\mathbb{F}\to\mathbb{P}) \qquad \mathbb{C} = 0{:}\mathbb{F}\&\mathbb{P} + 1{:}\delta\mathbb{C} \qquad \mathbb{F} = 2{:}\mathbb{P} + 3{:}\mathbb{F}\to\mathbb{P}\ .$$

Concretions of the form $(\boldsymbol{\nu}\tilde{\alpha})\langle V\rangle P$ correspond to terms of type $\mathbb{C}$; recursion on types is used to encode the tuple of restricted names $\tilde{\alpha}$. The function $[\![-]\!]_v$ translates values into the following terms of type $\mathbb{F}$:

$$[\![P]\!]_v = 2{:}[\![P]\!] \qquad [\![(x).P]\!]_v = 3{:}\lambda x.[\![P]\!]$$

while the function $[\![-]\!]$ translates processes into terms of type $\mathbb{P}$:

$$[\![V\bullet W]\!] = \tau{:}!(\pi_3[\![V]\!]_v)(\pi_2[\![W]\!]_v + \pi_3[\![W]\!]_v) \qquad [\![P \mid Q]\!] = [\![P]\!] \;||\; [\![Q]\!] \qquad [\![x]\!] = x \quad [\![\mathbf{0}]\!] = \mathbf{0}$$

$$[\![n(x).P]\!] = \mathsf{inp}{:}n\cdot!(\lambda x.[\![P]\!]) \qquad [\![\overline{n}(V)]\!] = \mathsf{out}{:}n\cdot!([\![V]\!]_v, [\![P]\!]) \qquad [\![(\boldsymbol{\nu}\alpha)P]\!] = \mathit{Res}\,\mathit{new}\alpha.[\![P]\!]\ .$$

The restriction map $\mathit{Res} : \delta\mathbb{P} \to \mathbb{P}$ filters the actions that a process emits, blocking actions that refer to the name that is being restricted. Output actions cause names to be extruded: the third summand records these names in the appropriate concretion.

$$\begin{array}{rcl} \mathit{Res}\ t & = & [t > \mathit{new}\alpha.\tau{:}!x[\alpha] \Rightarrow \tau{:}!\mathit{Res}\,x] \\ & + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{inp}{:}(\beta\cdot!x[\alpha]) \Rightarrow \mathsf{inp}{:}(\beta\cdot!\lambda y.\mathit{Res}\,(\mathit{new}\gamma.x[\gamma](y)))] \\ & + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{out}{:}(\beta\cdot!x[\alpha]) \Rightarrow \mathsf{out}{:}(\beta\cdot!3{:}x)] \end{array}$$

Parallel composition is a family of mutually dependent operations also including components such as $||_i$ of type $\mathbb{C}\&\mathbb{F}\to\mathbb{P}$ to say how values compose in parallel with concretions etc. All these components can be tupled together in a product and parallel composition defined as a simultaneous recursive definition:

— *Processes in parallel with processes:*

$$\begin{array}{rcl} t \;||\; u & = & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{out}{:}\beta\cdot!x \Rightarrow [u > \mathsf{inp}{:}\beta\cdot!y \Rightarrow \tau{:}!(x \;||\; y)]] \\ & + & \Sigma_{\beta\in\mathbb{N}}[u > \mathsf{inp}{:}\beta\cdot!y \Rightarrow \mathsf{inp}{:}\beta\cdot!(t \;||_a\; y)] \\ & + & \Sigma_{\beta\in\mathbb{N}}[u > \mathsf{out}{:}\beta\cdot!y \Rightarrow \mathsf{out}{:}\beta\cdot!(t \;||_c\; y)] \\ & + & [u > \tau{:}!y \Rightarrow \tau{:}!(t \;||\; y)] \\ & + & \text{symmetric cases} \end{array}$$

— *Concretions in parallel with values*

$$c \;||_i\; f \;=\; snd(\pi_0 c) \;||\; (\pi_3 f)(\pi_2(fst(\pi_0 c)) + \pi_3(fst(\pi_0 c))) + Res\,(new\alpha.(((\pi_1 c)[\alpha]) \;||_i\; f))$$

— *Concretions in parallel with processes*

$$c \;||_c\; t \;=\; 0{:}(fst(\pi_0 c), snd(\pi_0 c) \;||\; t) + 1{:}(new\alpha.((\pi_1 c)[\alpha] \;||_c\; t))$$

— *Values in parallel with processes*

$$f \;||_a\; t \;=\; \lambda x.(((\pi_3 f)x) \;||\; u)$$

The remaining cases are given symmetrically. The proposed encoding agrees with the reduction semantics of HOπ. The resulting bisimulation is analogous to the so called *higher-order bisimulation* [1, 14], and as such it is strictly finer than observational equivalence. It is an open problem whether it is possible to provide an encoding of HOπ that preserves and reflects the natural observational equivalence.

4.3 Mobile Ambients

We sketch an encoding of the mobility core of the Ambient Calculus, extending the encoding of Mobile Ambients with public names into HOPLA given in [7].

Again, we denote *name constants* with $a, b, \ldots$, *name variables* with $\alpha, \beta, \ldots$, and we let $n, m, \ldots$ range over both name constants and name variables. The terms of the language are constructed according the following grammar:

$$P \;::=\; \mathbf{0} \;\big|\; n[P] \;\big|\; P \mid P \;\big|\; (\boldsymbol{\nu}\alpha)P \;\big|\; \mathtt{in\_}n.P \;\big|\; \mathtt{out\_}n.P \;\big|\; \mathtt{open\_}n.P\ .$$

With respect to the HOPLA encoding, the syntax has been enriched with the restriction operator.

Types reflect the actions that ambient processes can perform, and can be given recursively by:

$$\begin{aligned}
\mathbb{P} \;&=\; \tau{:}!\mathbb{P} + \mathsf{in}{:}\mathbb{N}\otimes !\mathbb{P} + \mathsf{out}{:}\mathbb{N}\otimes !\mathbb{P} + \mathsf{open}{:}\mathbb{N}\otimes !\mathbb{P}\\
&\quad + \mathsf{mvin}{:}\mathbb{N}\otimes !\mathbb{C} + \mathsf{mvout}{:}\mathbb{N}\otimes !\mathbb{C}\\
&\quad + \overline{\mathsf{open}}{:}\mathbb{N}\otimes !\mathbb{P} + \overline{\mathsf{mvin}}{:}\mathbb{N}\otimes !\mathbb{F}\\
\mathbb{C} \;&=\; 0{:}\mathbb{P}\&\mathbb{P} + 1{:}\delta\mathbb{C}\\
\mathbb{F} \;&=\; \mathbb{P}\to\mathbb{P}
\end{aligned}$$

The injections in, out, and open correspond to the basic capabilities a process can exercise, while their action on the enclosing ambients is registered by the components mvin and mvout. The injections $\overline{\mathsf{open}}$ and $\overline{\mathsf{mvin}}$ record the receptive interactions that an ambient can (implicitly) have with the environment. Again, recursive types are used in concretions to record the sequence of names that must be extruded. In the HOPLA encoding, the type of concretions was $\mathbb{C} = \mathbb{P}\&\mathbb{P}$: it has been changed to reflect that mobility can cause extrusion of names, along the lines of our treatment of HOπ.[3]

[3]Note a minor notational change: the meaning of actions-coactions $\mathsf{mvin}, \overline{\mathsf{mvin}}$ and $\mathsf{open}, \overline{\mathsf{open}}$ has been swapped with respect to the encoding of MA into HOPLA.

The translation of terms is inherited from the HOPLA paper, with the addition of the rule for the restriction operator.

$$
\begin{array}{rclrcl}
[\![0]\!] & = & 0 & [\![\mathtt{in\_}n.P]\!] & = & \mathsf{in}\ n\cdot ![\![P]\!] \\
[\![n[P]]\!] & = & \mathit{Amb}\,(n, [\![P]\!]) & [\![\mathtt{out\_}n.P]\!] & = & \mathsf{out}\ n\cdot ![\![P]\!] \\
[\![P \mid Q]\!] & = & [\![P]\!] \ \ || \ \ [\![Q]\!] & [\![\mathtt{open\_}n.P]\!] & = & \mathsf{open}\ n\cdot ![\![P]\!] \\
[\![(\boldsymbol{\nu}\alpha)P]\!] & = & \mathit{Res}\,(\mathit{new}\alpha.[\![P]\!]) & & &
\end{array}
$$

The restriction map $\mathit{Res} : \delta\mathbb{P} \to \mathbb{P}$ filters the actions that a process emit, and blocks actions that refer to the name that is restricted. In fact, in Pure Mobile Ambients, the only scope extrusions are caused by mobility, and not by pre-actions.

$$
\begin{array}{rcl}
\mathit{Res} & : & \delta\mathbb{P} \to \mathbb{P} \\
\mathit{Res}\ t & = & [t > \mathit{new}\alpha.\tau{:}x[\alpha] \Rightarrow \tau{:}\mathit{Res}\,x)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{in}{:}(\beta\cdot !x[\alpha]) \Rightarrow \mathsf{in}{:}(\beta\cdot !\mathit{Res}\,x)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{out}{:}(\beta\cdot !x[\alpha]) \Rightarrow \mathsf{out}{:}(\beta\cdot !\mathit{Res}\,x)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{open}{:}(\beta\cdot !x[\alpha]) \Rightarrow \mathsf{open}{:}(\beta\cdot !\mathit{Res}\,x)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{mvin}{:}(\beta\cdot !x[\alpha]) \Rightarrow \mathsf{mvin}{:}(\beta\cdot !1{:}x) \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\mathsf{mvout}{:}(\beta\cdot !x[\alpha]) \Rightarrow \mathsf{mvout}{:}(\beta\cdot !1{:}x) \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\overline{\mathsf{open}}{:}(\beta\cdot !x[\alpha]) \Rightarrow \overline{\mathsf{open}}{:}(\beta\cdot !\mathit{Res}\,x)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathit{new}\alpha.\overline{\mathsf{mvin}}{:}(\beta\cdot !x[\alpha]) \Rightarrow \overline{\mathsf{mvin}}{:}(\beta\cdot !\lambda y.\mathit{Res}\,(\mathit{new}\gamma.x[\gamma](y)))]
\end{array}
$$

Parallel composition is a family of operations, one of which is a binary operation between processes, $||_{\mathbb{P}\&\mathbb{P}}$: $\mathbb{P}\&\mathbb{P} \to \mathbb{P}$. The family is defined in a simultaneous recursive definition below.

- Processes in parallel with processes:

$$
\begin{array}{rcl}
t \ || \ u & = & \Sigma_{\beta\in\mathbb{N}}[t > \overline{\mathsf{open}}{:}\ \beta\cdot !x \Rightarrow [u > \mathsf{open}{:}\ \beta\cdot !y \Rightarrow \tau{:}\cdot!(x \ || \ y)]] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \overline{\mathsf{mvin}}{:}\ \beta\cdot !f \Rightarrow [u > \mathsf{mvin}{:}\ \beta\cdot !c \Rightarrow \tau{:}\cdot!(c \ || \ f)]] \\
& + & [t > \tau{:}!x \Rightarrow \tau{:}!(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{in}{:}\beta\cdot !x \Rightarrow \mathsf{in}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{out}{:}\beta\cdot !x \Rightarrow \mathsf{out}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{open}{:}\beta\cdot !x \Rightarrow \mathsf{open}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{mvin}{:}\beta\cdot !x \Rightarrow \mathsf{mvin}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{mvout}{:}\beta\cdot !x \Rightarrow \mathsf{mvout}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \overline{\mathsf{open}}{:}\beta\cdot !x \Rightarrow \overline{\mathsf{open}}{:}\beta\cdot !(x \ || \ u)] \\
& + & \Sigma_{\beta\in\mathbb{N}}[t > \overline{\mathsf{mvin}}{:}\beta\cdot !x \Rightarrow \overline{\mathsf{mvin}}{:}\beta\cdot !(x \ || \ u)] \\
& + & \text{symmetric cases.}
\end{array}
$$

  All summands except the first two correspond to congruence rules.

- Concretions in parallel with abstractions:

$$c \ ||_i \ f = \mathit{snd}(\pi_0 c) \ \ || \ \ f(\mathit{fst}(\pi_0 c)) + \mathit{Res}\,(\mathit{new}\alpha.(((\pi_1 c)[\alpha]) \ \ ||_i \ \ f))$$

- Concretions in parallel with processes:

$$c \ ||_c \ t = 0{:}(fst(\pi_0 c), snd(\pi_0 c) \ || \ t) + 1{:}(new\alpha.((\pi_1 c)[\alpha] \ ||_c \ t))$$

- Abstractions in parallel with processes:

$$f \ ||_a \ t = \lambda x.((fx) \ || \ u)$$

Remaining cases are given symmetrically.

Finally, ambient creation can be defined recursively in new-HOPLA as an operation $Amb : \mathbb{N}\&\mathbb{P} \to \mathbb{P}$:

$$\begin{array}{rcl} Amb\,(m,t) & = & [t > \tau{:}!x \Rightarrow \tau{:}!Amb\,(m,x)] \\ & + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{in}{:}\beta\cdot !x \Rightarrow \mathsf{mvin}{:}\beta\cdot !(Amb\,(m,x),0) \\ & + & \Sigma_{\beta\in\mathbb{N}}[t > \mathsf{out}{:}\beta\cdot !x \Rightarrow \mathsf{mvout}{:}\beta\cdot !(Amb\,(m,x),0) \\ & + & [t > \mathsf{mvout}{:}m\cdot !c \Rightarrow \tau{:}!Extr\,(m,c)] \\ & + & \overline{\mathsf{open}}{:}m\cdot !t + \overline{\mathsf{mvin}}{:}m\cdot !\lambda y.Amb\,(m,t \ || \ y) \end{array}$$

where the map $Extr : \mathbb{N}\&\mathbb{C} \to \mathbb{P}$ extrudes names across ambient's boundary after an mvout action:

$$Extr(m,c) \ = \ fst(\pi_0 c) \ || \ Amb\,(m, snd(\pi_0 c)) + Res\,(new\alpha.(Extr\,(m,(\pi_1 c)[\alpha]))) \ .$$

The denotations of ambients are determined by their capabilities: an ambient $m[t]$ can perform the internal (τ) actions of t, enter a brother ambient ($\mathsf{mvin}\ n$) if called upon to do so by an $\mathsf{in}\ n$ action of t, exit its parent ambient ($\mathsf{mvout}\ n$) if called upon to do so by an $\mathsf{out}\ n$ action of t, be exited if t so requests through an $\mathsf{mvout}\ m$ action, be opened ($\overline{\mathsf{open}}\ m$), or be entered by an ambient ($\overline{\mathsf{mvin}}\ m$); other pre-actions are restricted away. The tree-containment structure of ambients is captured in the chain of $\overline{\mathsf{open}}\ m$'s that they can perform.

The semantics given above agrees with the reduction semantics of Mobile Ambients, possibly up to structural congruence. It is easy to prove that if the encodings of two processes are bisimilar in new-HOPLA, then they are *strong* reduction barbed congruent. We do not know if the converse is true (it is false if we add recursive definition of processes to the fragment of MA we consider). Yet, in MA, strong reduction barbed congruence is an extremely discriminating relation and its interest is very limited.

References

[1] G. Boudol. Towards a lambda calculus for concurrent and communicating systems. In *Proc. TAPSOFT '89*, volume 351 of *LNCS*, pages 149–161. Springer Verlag, 1989.

[2] G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the π-calculus. In *Proc. CTCS'97*, volume 1290 of *LNCS*. Springer Verlag, 1997.

[3] M. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *Proc. 11th LICS*. IEEE Computer Society Press, 1996.

[4] J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[5] A. D. Gordon. Bisimilarity as a theory of functional programming: mini-course. Notes Series BRICS-NS-95-3, BRICS, Department of Computer Science, University of Aarhus, July 1995.

[6] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[7] M. Nygaard and G. Winskel. Hopla—a higher-order process language. In *Proc. CONCUR'02*, volume 2421 of *LNCS*. Springer Verlag, 2002.

[8] M. Nygaard and G. Winskel. Domain theory for concurrency. To appear in *Theoretical Computer Science*, special issue on domain theory, 2003.

[9] M. Nygaard and G. Winskel. Full abstraction for HOPLA. In *Proc. CONCUR'03*, LNCS. Springer Verlag, 2003.

[10] A. M. Pitts. Operationally-based theories of program equivalence. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, pages 241–298. Cambridge University Press, 1997.

[11] A. M. Pitts and M. J. Gabbay. A metalanguage for programming with bound names modulo renaming. In *Proc. MPC 2000*, volume 1837 of *LNCS*. Springer Verlag, 2000.

[12] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[13] I. Stark. A fully-abstract domain model for the π-calculus. In *Proc. 11th LICS*. IEEE Computer Society Press, 1996.

[14] B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Department of Computing, Imperial College, 1990.

[15] F. Zappa Nardelli. *De la sémantique des processus d'ordre supérieur*. PhD thesis, Université de Paris 7, 2003. Available in English from `http://www.di.ens.fr/~zappa`.

A Proofs

A.1 Proofs from Section 2

Proof of Lemma 2.2, page 10 By rule induction.

Inaction. Suppose $A; x{:}\mathbb{Q}, \Gamma; d \vdash 0 : \mathbb{P}$. As $0[t/x] = 0$, we derive $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash 0[t/x] : \mathbb{P}$ by the rule for inactive process.

Variable. Suppose $A; x{:}\mathbb{Q}, \Gamma; d \vdash y : \mathbb{P}$. If $y \neq x$ then $y[t/x] = y$ and we derive $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash y[t/x] : \mathbb{P}$ by the rule for identity. If $y = x$, then $y[t/x] = t$ and $\mathbb{P} = \mathbb{Q}$: we derive and $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash y[t/x] : \mathbb{P}$ from $A'; \Gamma'; d' \vdash t : \mathbb{Q}$ by weakening.

Weakening. Suppose that $A''; x{:}\mathbb{Q}, \Gamma''; d'' \vdash u : \mathbb{P}$ has been derived from $A; \Gamma; d \vdash u : \mathbb{P}$ for $A'' \supseteq A, \Gamma'' \supseteq \Gamma, d'' \supseteq d$. As $\Gamma'' \cup \Gamma'$ is consistent, then $\Gamma \cup \Gamma'$ is consistent as well. If $x{:}\mathbb{Q} \in \Gamma$, then by the induction hypothesis we have $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{P}$ where $\overline{d} = (d \setminus x) \cup d' \cup \{\{\alpha\} \times (A' \cup \Gamma') \mid (\alpha, x) \in d\}$. By weakening we conclude $A'' \cup A'; \Gamma'' \cup \Gamma'; (d'' \setminus x) \cup d' \cup \{\{\alpha\} \times (A' \cup \Gamma') \mid (\alpha, x) \in d''\} \vdash u[t/x] : \mathbb{P}$. If $x{:}\mathbb{Q} \notin \Gamma$, then $x \notin \mathrm{fv}(u)$ and $u[t/x] = u$. The result follows from $A; \Gamma; d \vdash u : \mathbb{P}$ by weakening.

Tensor. Suppose that $A; x{:}\mathbb{Q}; d \vdash \alpha \cdot u : \mathbb{P}$ has been derived from $A; x{:}\mathbb{Q}; d \vdash u : \mathbb{P}$ for $\alpha \in A$. By the induction hypothesis we have $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash \alpha \cdot u[t/x] : \mathbb{P}$. As $\alpha \in A$, $\alpha \in A \cup A'$ as well, and we conclude by the rule for tensor.

Name projection. Analogous to tensor.

Name abstraction. Suppose that $A; x{:}\mathbb{Q}, \Gamma; d \vdash \lambda\alpha.u : \mathbb{N} \to \mathbb{P}$ has been derived from $\alpha{:}\mathbb{N}, A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{P}$ with $\alpha \notin d$. By the induction hypothesis we get $(\alpha{:}\mathbb{N}, A) \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{P}$. As substitution is capture-avoiding, $\alpha{:}\mathbb{N} \notin A'$, and $(\alpha{:}\mathbb{N}, A) \cup A' = \alpha{:}\mathbb{N}, (A \cup A')$. It remains to show that $\alpha \notin \overline{d}$. But $\alpha \notin d'$ because $\alpha \notin A'$, and $\alpha \notin d$ because of the hypothesis. Thus, we can conclude by the rule for name abstraction.

Name application. Suppose that $A; x{:}\mathbb{Q}, \Gamma; d \vdash u\alpha : \mathbb{P}$ has been derived from $A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{N} \to \mathbb{P}$, with $\alpha \in A$. By the induction hypothesis we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{N} \to \mathbb{P}$. As $\alpha \in A$, $\alpha \in (A \cup A')$. As $u[t/x]\alpha = u\alpha[t/x]$, by the rule for name application we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u\alpha[t/x] : \mathbb{P}$, as required.

New name abstraction. Suppose that $A; x{:}\mathbb{Q}, \Gamma; (d \setminus \alpha) \vdash \mathit{new}\alpha.u : \delta\mathbb{P}$ has been derived from $\alpha{:}\mathbb{N}, A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{P}$ with $\alpha \notin d$. By the induction hypothesis we get $(\alpha{:}\mathbb{N}, A) \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{P}$. As substitution is capture-avoiding, $\alpha{:}\mathbb{N} \notin A'$, and $(\alpha{:}\mathbb{N}, A) \cup A' = \alpha{:}\mathbb{N}, (A \cup A')$. It also holds $\overline{d} \setminus \alpha = ((d \setminus x) \cup d' \cup \{(\alpha, z) \mid z \in \Gamma' \text{ and } (\alpha, x) \in d\} \cup \{(\alpha, \beta) \mid \beta \in A' \text{ and } (\alpha, x) \in d\}) \setminus \alpha = ((d \setminus \alpha) \setminus x) \cup d' \cup \{(\alpha, z) \mid z \in \Gamma' \text{ and } (\alpha, x) \in (d \setminus \alpha)\} \cup \{(\alpha, \beta) \mid \beta \in A' \text{ and } (\alpha, x) \in (d \setminus \alpha)\}$. Then we can conclude by the rule for new name abstraction.

New name application. Suppose that $\alpha{:}\mathbb{N}, A; x{:}\mathbb{Q}, \Gamma; d \cup (\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A)) \vdash u[\alpha] : \mathbb{P}$ has been derived $A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \delta\mathbb{P}$. We want to show that $\alpha{:}\mathbb{N}, (A \cup A'); \Gamma \cup \Gamma'; \overline{d} \vdash u[\alpha][t/x] : \mathbb{P}$ where $\overline{d}$ is the set of distinctions:

$$((d \cup (\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A))) \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (d \cup (\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A)))\}\ .$$

By the induction hypothesis we get $A \cup A'; \Gamma \cup \Gamma'; (d \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in d\} \vdash u[t/x] : \delta\mathbb{P}$. As $(\alpha, x) \in (d \cup (\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A)))$, the hypothesis guarantee that $\alpha \notin A'$. By the rule for new name application we derive $\alpha{:}\mathbb{N}, (A \cup A'); \Gamma \cup \Gamma'; d_1 \vdash u[t/x][\alpha] : \mathbb{P}$, where the set of distinctions d_1 is

$$(d \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in d\} \cup (\{\alpha\} \times (\Gamma \cup A))\ .$$

Now, $u[t/x][\alpha] = u[\alpha][t/x]$. A little care is needed to show that d_1 is equal to $\overline{d}$. The set $\overline{d}$ can be rewritten as

$$\begin{aligned}&(d \setminus x) \cup ((\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A)) \setminus x) \cup d' \\ &\cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in d\} \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\{\alpha\} \times ((x{:}\mathbb{Q}, \Gamma) \cup A))\}\end{aligned}$$

and in turn as

$$(d \setminus x) \cup (\{\alpha\} \times (\Gamma \cup A)) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in d\} \cup (\{\alpha\} \times (A' \cup \Gamma'))$$

and this is equal to d_1, as required.

Process abstraction. Suppose that $A; x{:}\mathbb{Q}, \Gamma; d \vdash \lambda y.u : \mathbb{R} \to \mathbb{P}$ has been derived from $A; y{:}\mathbb{R}, x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{P}$ with $y \notin d$. By the induction hypothesis we get $A \cup A'; (y{:}\mathbb{R}, \Gamma) \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{P}$. As substitution is capture-avoiding, $y{:}\mathbb{R} \notin \Gamma'$, and $(x{:}\mathbb{Q}, \Gamma) \cup \Gamma' = x{:}\mathbb{Q}, (\Gamma \cup \Gamma')$. It remains to show that $x \notin \overline{d}$. But $x \notin d'$ because $x \notin \Gamma'$, and $x \notin d$ because of the hypothesis. Thus, we can conclude by the rule for process abstraction.

Process application. Suppose that $A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{P}$ has been derived from $A; x{:}\mathbb{Q}, \Gamma; d \vdash u : \mathbb{R} \to \mathbb{P}$ and $A; x{:}\mathbb{Q}, \Gamma; d \vdash v : \mathbb{R}$. By the induction hypothesis we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u[t/x] : \mathbb{R} \to \mathbb{P}$ and $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash v[t/x] : \mathbb{R}$. As $u[t/x]v[t/x] = uv[t/x]$, by the rule for process application we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash uv[t/x] : \mathbb{P}$, as required.

Recursion. Analogous to process abstraction.

Sum. Suppose that $A; x{:}\mathbb{Q}, \Gamma; d \vdash \Sigma_{i \in I} u_i : \mathbb{P}$ has been derived from $A; x{:}\mathbb{Q}, \Gamma; d \vdash u_i : \mathbb{P}$ for all $i \in I$. By the induction hypothesis we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \vdash u_i[t/x] : \mathbb{P}$ for all $i \in I$. As $\Sigma_{i \in I}(u_i[t/x]) = \Sigma_{i \in I} u_i[t/x]$, by the rule for sum we get $A \cup A'; \Gamma \cup \Gamma'; \overline{d} \Sigma_{i \in I} \vdash u_i[t/x] : \mathbb{P}$, as required.

Injection, projection, lifting, recursive type. Straightforward use of the induction hypothesis, along the lines of the case of tensor.

Pattern matching. A little care is required to avoid getting confused by the type environments and the sets of distinctions. We detail the case in which the variable x belongs both to the type environment of the matched term, and to the type environment of the continuation. The other cases are simpler.

Suppose that

$$B \cup B; (x{:}\mathbb{Q}, \Delta) \cup (x{:}\mathbb{Q}, \Delta'); d \vdash [w > p(y) \Rightarrow v] : \mathbb{P}$$

has been derived from

$$B'; x{:}\mathbb{Q}, \Delta'; \delta' \vdash w : \mathbb{R} \qquad B'; x{:}\mathbb{Q}, \Delta'; \delta'; ; y{:}\mathbb{S} \vdash p : \mathbb{R} \qquad B; y{:}\mathbb{S}, x{:}\mathbb{Q}, \Delta; \delta \vdash v{:}\mathbb{P}$$

where

$$d = (\delta \setminus y) \cup \delta' \cup \{\{\beta\} \times (B' \cup (\Delta', x{:}\mathbb{Q})) \mid (\beta, y) \in \delta\}$$

and

$$B' \cap \{\beta \mid (\beta, y) \in \delta\} \ = \ \emptyset \ . \tag{1}$$

With respect to the hypothesis of the Lemma, $A = B \cup B'$ and $x{:}\mathbb{Q}, \Gamma = (x{:}\mathbb{Q}, \Delta) \cup (x{:}\mathbb{Q}, \Delta')$. Also remember that $A'; \Gamma'; d' \vdash t{:}\mathbb{Q}$ and

$$A' \cap \{\beta \mid (\beta, x) \in d\} = \emptyset \ . \tag{2}$$

We want to show that $B \cup B' \cup A'; \Delta \cup \Delta' \cup \Gamma'; \overline{d} \vdash [w > p(y) \Rightarrow v][t/x] : \mathbb{P}$ where $\overline{d} = (d \setminus x) \cup d' \cup \{\beta \times (A', \Gamma') \mid (\beta, x) \in d\}$.

By the induction hypothesis we have:

$$B' \cup A'; \Delta' \cup \Gamma'; d_1 \vdash w[t/x] : \mathbb{R} \qquad B' \cup A'; \Delta' \cup \Gamma'; d_1; ; y{:}\mathbb{S} \vdash p[t/x] : \mathbb{R}$$

where

$$d_1 \ = \ (\delta' \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \ .$$

We also have:

$$B \cup A'; (y{:}\mathbb{S}, \Delta) \cup \Gamma'; d_2 \vdash v[t/x] : \mathbb{P}$$

where

$$d_2 \;=\; (\delta \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta\} \;.$$

We show now that $(B' \cup A') \cap \{\gamma \mid (\gamma, y) \in d_2\} = \emptyset$. For that, we rewrite the set $\{\gamma \mid (\gamma, y) \in d_2\}$ can be rewritten as the union of the sets S_i defined as

$$S_1 = \{\gamma \mid (\gamma, y) \in (\delta \setminus x)\} \qquad S_2 = \{\gamma \mid (\gamma, y) \in d'\}$$
$$S_3 = \{\gamma \mid (\gamma, y) \in \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta\}\} \;.$$

Now, $B' \cap S_1 = \emptyset$ because of hypothesis (1), and $A' \cap S_1 = \emptyset$ because of (2). This implies $(A' \cup B') \cap S_1 = \emptyset$. The sets S_2 and S_3 are empty because y is a bound variable and without loss of generality we can suppose $y \notin \Gamma'$.

Then, we can apply the pattern matching typing rule to obtain

$$B' \cup A' \cup B; \Delta' \cup \Gamma' \cup \Delta; d_3 \vdash [w[t/x] > p[t/x](y) \Rightarrow v[t/x]] : \mathbb{P}$$

where

$$d_3 = (d_2 \setminus y) \cup d_1 \cup \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in d_2\} \;.$$

By definition of substitution $[w[t/x] > p[t/x](y) \Rightarrow v[t/x]] = [w > p(y) \Rightarrow v][t/x]$. It remains to prove that $d_3 = \overline{d}$. With a lot of patience, we see that

$$\begin{aligned}
\overline{d} &= (d \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in d\} \\
&= (((\delta \setminus y) \cup \delta' \cup \{\{\beta\} \times (B' \cup (\Delta', x{:}\mathbb{Q})) \mid (\beta, y) \in \delta\}) \setminus x) \cup d' \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\delta \setminus y)\} \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \{\gamma \times (B' \cup (\Delta', x{:}\mathbb{Q})) \mid (\gamma, y) \in \delta\}\} \\
&= (\delta \setminus y \setminus x) \cup (\delta' \setminus x) \cup \{\beta \times (B' \cup \Delta') \mid (\beta, y) \in \delta\} \cup d' \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\delta \setminus y)\} \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, y) \in \delta\} \\
&= (\delta \setminus y \setminus x) \cup (\delta' \setminus x) \cup d' \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\delta \setminus y)\} \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \cup \\
&\qquad \{\{\beta\} \times (B' \cup \Delta' \cup A' \cup \Gamma') \mid (\beta, y) \in \delta\}
\end{aligned}$$

In the last expression, the set $\{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\delta \setminus y)\}$ can be rewritten as $\{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in (\delta \setminus y)\}$. With more patience, we calculate:

$$\begin{aligned}
d_3 &= (d_2 \setminus y) \cup d_1 \cup \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in d_2\} \\
&= (((\delta \setminus x) \cup d' \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta\}) \setminus y) \cup (\delta' \setminus x) \cup d' \cup \\
&\qquad \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \cup \\
&\qquad \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in (\delta \setminus x)\} \cup \\
&\qquad \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in d'\} \\
&\qquad \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in \{\gamma \times (A' \cup \Gamma') \mid (\gamma, x) \in \delta\}\}
\end{aligned}$$

As $(d' \setminus y) \subseteq d'$, and since y is a bound variable an without loss of generality we can suppose $y \notin \Gamma'$ (and in turn $y \notin d'$), we have

$$\begin{aligned} d_3 = (\delta \setminus x \setminus y) \cup (\delta' \setminus x) \cup d' \cup \\ \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta\} \cup \{\{\beta\} \times (A' \cup \Gamma') \mid (\beta, x) \in \delta'\} \cup \\ \{\{\beta\} \times (B' \cup A' \cup \Delta' \cup \Gamma') \mid (\beta, y) \in (\delta \setminus x)\}\ . \end{aligned}$$

We conclude that $\overline{d} = d_3$ because they are union of the same subsets.

Typing rules for patterns follow along the same lines (they are easier). The induction is now complete. □

Proof of Theorem 2.4, page 11 By rule induction on the derivation of $\mathbb{P}; s \vdash t \xrightarrow{p\ (x)} u$.

Prefixing. Suppose that $!\mathbb{P}; s \vdash !t \xrightarrow{!x\ (x)} t$. Then $s \vdash !t : !\mathbb{P}$ and so by the typing rules, $s \vdash t : \mathbb{P}$ as wanted.

Process abstraction. Suppose that $\mathbb{P} \to \mathbb{Q}; \lambda y.t \xrightarrow{u \mapsto p\ (x)} t'$ with $s;; x{:}\mathbb{R} \vdash u \mapsto p : \mathbb{P} \to \mathbb{Q}$. By typing of patterns, we have $s \vdash u : \mathbb{P}$ and $s;; x{:}\mathbb{R} \vdash p : \mathbb{Q}$. The induction hypothesis then yields $s \vdash t' : \mathbb{R}$ as wanted. Note that the substitution $t[u/x]$ is well-formed because $A; y{:}\mathbb{Q}; A \vdash t : \mathbb{P}$ follows from $s \vdash \lambda y.t : \mathbb{P} \to \mathbb{Q}$, for a bijection $\sigma : s \to A$.

Process application. Suppose that $\mathbb{Q}; s \vdash tu \xrightarrow{p\ (x)} t'$ has been derived from $\mathbb{P} \to \mathbb{Q} : s \vdash t \xrightarrow{u \mapsto p(x)} t'$ with $s;; x{:}\mathbb{R} \vdash p(x) : \mathbb{Q}$. By the premise and the typing rules, we have $s \vdash t : \mathbb{P} \to \mathbb{Q}$ and $s \vdash u : \mathbb{P}$, such that $s;; x{:}\mathbb{R} \vdash u \mapsto p : \mathbb{P} \to \mathbb{Q}$. The induction hypothesis then yields $s \vdash t' : \mathbb{R}$ as wanted.

New name abstraction. Suppose that $\delta\mathbb{P}; s \vdash new\alpha.t \xrightarrow{new\alpha.p[x'[\alpha]/x]\ (x')} new\alpha.t'$ has been derived from $\mathbb{P}; s \dot\cup \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha]\ (x)} t'[a/\alpha]$ with $s;; x'{:}\delta\mathbb{R} \vdash new\alpha.p[x'[\alpha]/x] : \delta\mathbb{P}$. By the typing rules, we get $A, \alpha{:}\mathbb{N}; \emptyset; A, d \vdash t : \mathbb{P}$, for $\sigma : s + \{a\} \to A + \{\alpha\}$. This implies $s \dot\cup \{a\} \vdash t[a/\alpha] : \mathbb{P}$. Again, by typing rules we get $A, \alpha{:}\mathbb{N}; \emptyset; A, d;; x{:}\mathbb{R} \vdash p : \mathbb{P}$ and then $s \dot\cup \{a\};; x{:}\mathbb{R} \vdash p[a/\alpha] : \mathbb{P}$. By the induction hypothesis we have $s \dot\cup \{a\} \vdash t'[a/\alpha] : \mathbb{R}$. This implies that $A, \alpha{:}\mathbb{N}; \emptyset; A, \alpha \vdash t' : \mathbb{R}$ and we get $s \vdash new\alpha.t' : \delta\mathbb{R}$ by the typing rules, as wanted.

New name application. Suppose that $\mathbb{P}; s \dot\cup \{a\} \vdash t[a] \xrightarrow{p[a/\alpha]\ (x)} t'[a]$ has been derived from $\delta\mathbb{P}; s \vdash t \xrightarrow{new\alpha.p[x'[\alpha]/x]\ (x')} t'$, with $s \dot\cup \{a\};; x{:}\mathbb{R} \vdash p[a/\alpha] : \mathbb{P}$. By the typing rules we get $s \vdash t : \delta\mathbb{P}$. Also, as the pattern is well-typed, $A, \alpha{:}\mathbb{N}; \emptyset; A, \alpha;; x{:}\mathbb{R} \vdash p : \mathbb{P}$ must hold for $\sigma : s + \{a\} \to A + \{\alpha\}$. By the typing rules we get $s;; x'{:}\delta\mathbb{R} \vdash new\alpha.p[x'[\alpha]/x] : \delta\mathbb{P}$. By the induction hypothesis, $s \vdash t' : \delta\mathbb{R}$, and by the typing rules we conclude $s \dot\cup \{a\} \vdash t'[a] : \mathbb{R}$.

Pattern matching. Suppose that $\mathbb{Q}; s \vdash [t > p(y) \Rightarrow u] \xrightarrow{q\ (x)} u'$ has been derived from $\mathbb{P}; s \vdash t \xrightarrow{p(y)} t'$ and $\mathbb{Q}; s \vdash u[t'/y] \xrightarrow{q(x)} u'$ with $s;; y{:}\mathbb{S} \vdash p : \mathbb{P}$ and $s;; x{:}\mathbb{R} \vdash q : \mathbb{Q}$. By the induction hypothesis applied to the first premise, we get $s \vdash t' : \mathbb{S}$. Thus, as $A; x{:}\mathbb{R}; A \vdash u : \mathbb{Q}$ for $\sigma : s \to A$, the substitution $u[t'/x]$ is well-formed. By the induction hypothesis applied to the second premise, we get $s \vdash u' : \mathbb{R}$, as wanted.

The remaining cases are handled similarly. □

A.2 Proofs from Section 3

To prove that bisimilarity is a congruence relation we need some auxiliary lemmas and definitions.

Lemma A.1 *If $s \vdash t \sim u : \mathbb{P}$, then for all $s' \supseteq s$ it holds $s' \vdash t \sim u : \mathbb{P}$.*

Proof Suppose $s \vdash t \sim u : \mathbb{P}$ and $s' \supseteq s$. We want to show that $s' \vdash t \sim u : \mathbb{P}$. For that let $s'' \supseteq s'$ and suppose $s'' \vdash t \xrightarrow{p} t'$. As $s \vdash t \sim u : \mathbb{P}$ and $s'' \supseteq s$, there exists a term u' such that $s'' \vdash u \xrightarrow{p} u'$ and $s'' \vdash t' \sim u'$, as required. The symmetric case is handled similarly. $\square$

Lemma A.2 *Bisimilarity is an equivalence relation.*

Proof It is easy to see that bisimilarity is reflexive and symmetric. For transitivity, suppose $s \vdash t \sim u : \mathbb{P}$ and $s \vdash u \sim v : \mathbb{P}$. We want to show that $s \vdash t \sim v : \mathbb{P}$. Let $s' \supseteq s$ and suppose $s' \vdash t \xrightarrow{p} t'$. As $s \vdash t \sim u : \mathbb{P}$, there exists a term u' such that $s' \vdash u \xrightarrow{t} u'$ and $s' \vdash t' \sim u' : \mathbb{P}$. In turn, as $s \vdash u \sim v : \mathbb{P}$, there exists a term v' such that $s' \vdash v \xrightarrow{t} v'$ and $s' \vdash u' \sim v' : \mathbb{P}$. The result follows by coinduction. $\square$

Definition A.3 (Closure) *A* $(A; \Gamma; d)$-closure *is a triple $(s, \rho, [\vec{u}/\vec{x}])$, where*

1. *s is a set of names;*
2. *$\rho : A \to s$ is a map such that $\rho(\alpha) \neq \rho(\beta)$ whenever $(\alpha, \beta) \in d$, and $\rho(\alpha) \notin \mathsf{n}(u_i)$ whenever $(\alpha, u_i) \in d$ (that is, ρ substitutes name constants in s for name variables in A and respects distinctions);*
3. *$[\vec{u}/\vec{x}]$ is a substitution assigning closed terms to the process variables in Γ such that $s \vdash u_i : \Gamma(x_i)$ for all i.*

Closures are ranged over by Ξ. Given a type judgement $A; \Gamma; d \vdash t : \mathbb{P}$ and a $(A; \Gamma; d)$-closure $\Xi = (s, \rho, [\vec{u}/\vec{x}])$, we write $t[\Xi]$ for the term $\rho t[\vec{u}/\vec{x}]$ and s_Ξ for s. Remark that $s \vdash \rho t[\vec{u}/\vec{x}] : \mathbb{P}$ is a valid type judgement.

Definition A.4 (Open extension) *If $\mathcal{R}$ relates closed terms, we write $\mathcal{R}^\circ$ for its open extension, relating $A; \Gamma; d \vdash t : \mathbb{P}$ and $A; \Gamma; d \vdash u : \mathbb{P}$ if $s_\Xi \vdash t[\Xi] \mathrel{\mathcal{R}} u[\Xi] : \mathbb{P}$ holds for all $(A; \Gamma; d)$-closures Ξ.*

We write $\mathcal{R}_c$ for the restriction of a type-respecting relation to closed terms. For a type-respecting relation $\mathcal{R}$ we write $\mathcal{R}$ also for the relation induced on actions, given inductively by

$$\frac{A; \Gamma; d; ; x{:}\mathbb{R} \vdash p \mathrel{\mathcal{R}} q : \mathbb{P}}{A'; \Gamma'; d'; ; x{:}\mathbb{R} \vdash p \mathrel{\mathcal{R}} q : \mathbb{P}} \begin{array}{c} A \subseteq A' \\ \Gamma \subseteq \Gamma' \\ d \subseteq d' \end{array} \qquad \frac{}{E; ; x{:}\mathbb{R} \vdash !x \mathrel{\mathcal{R}} !x : !\mathbb{R}}$$

$$\frac{A; \Gamma; d; ; x{:}\mathbb{R} \vdash p \mathrel{\mathcal{R}} q : \mathbb{P} \quad \alpha \in A}{A; \Gamma; d; ; x{:}\mathbb{R} \vdash \alpha \cdot p \mathrel{\mathcal{R}} \alpha \cdot q : \mathbb{N} \otimes \mathbb{P}} \qquad \frac{E; ; x{:}\mathbb{R} \vdash p \mathrel{\mathcal{R}} q : \mathbb{P}_j \quad j \in I}{E; ; x{:}\mathbb{R} \vdash j{:}p \mathrel{\mathcal{R}} j{:}q : \Sigma_{i \in I} \mathbb{P}_i}$$

$$\frac{A;\Gamma;d;;x{:}\mathbb{R}\vdash p\,\mathcal{R}\,q:\mathbb{P}\quad \alpha\in A}{A;\Gamma;d;;x{:}\mathbb{R}\vdash \alpha\mapsto p\,\mathcal{R}\,\alpha\mapsto q:\mathbb{P}}\qquad \frac{A;\Gamma;d\vdash u\,\mathcal{R}\,v:\mathbb{P}\qquad A;\Gamma;d;;x{:}\mathbb{R}\vdash p\,\mathcal{R}\,q:\mathbb{Q}}{A;\Gamma;d;;x{:}\mathbb{R}\vdash u\mapsto p\,\mathcal{R}\,v\mapsto q:\mathbb{P}}$$

$$\frac{\alpha:\mathbb{N},A;\Gamma;d;;x{:}\mathbb{R}\vdash p\,\mathcal{R}\,q:\mathbb{P}}{A;\Gamma;(d\setminus\alpha);;x'{:}\delta\mathbb{R}\vdash \mathit{new}\alpha.p[x'[\alpha]/x]\;\mathcal{R}\;\mathit{new}\alpha.q[x'[\alpha]/x]:\delta\mathbb{P}}$$

The open extension of $\sim$ is closed under weakening:

Lemma A.5 *If $A;\Gamma;d\vdash t\sim^\circ u:\mathbb{P}$, then for all $A'\supseteq A,\Gamma'\supseteq\Gamma,d'\supseteq d$ we have $A';\Gamma';d'\vdash t\sim^\circ u:\mathbb{P}$.*

Proof Follows from the definition of open extension and from Lemma A.1. $\square$

Some terminology: a type respecting relation is said *operator respecting* if it is preserved by all the operators of the language. A *congruence* is an operator respecting relation that is also an equivalence.

Following Howe, we define an auxiliary relation, called the *precongruence candidate*, that, by construction, contains $\sim^\circ$ and is operator preserving. In what follows, we omit the $\mathbb{N}$ type in the environment of name variables, and we occasionally use E as a concise abbreviation for a typing environment $A;\Gamma;d$. Also, when no ambiguity arises, we use commas to denote the disjoint union of sets.

Definition A.6 (The precongruence candidate) *The* precongruence candidate*, $\hat{\sim}$, is the smallest type-respecting relation closed under the rules reported in Figure 1.*

We need several technical lemmas. The lemma below is fundamental to show that in a derivation of $s\vdash \mathit{new}\alpha.t\;\hat{\sim}_c\;u:\delta\mathbb{P}$ the choice of the name a used to derive $s\,\dot{\cup}\,\{a\}\vdash t[a/\alpha]\;\hat{\sim}_c\;v:\mathbb{P}$ is irrelevant, as far as it is fresh.

Lemma A.7 *If there is a derivation of $A,\alpha;\Gamma;d,(\{\alpha\}\times(A\cup\Gamma))\vdash t\;\hat{\sim}\;u:\mathbb{P}$, then there is a derivation of the same height of $A,\beta;\Gamma;d,(\{\beta\}\times(A,\Gamma))\vdash t[\beta/\alpha]\;\hat{\sim}\;u[\beta/\alpha]:\mathbb{P}$.*

We then prove that the precongruence candidate is closed under substitutions.

Lemma A.8

1. *if $A;\Gamma,x{:}\mathbb{Q};d\vdash u\;\hat{\sim}\;u':\mathbb{P}$ and $A';\Gamma';d'\vdash t\;\hat{\sim}\;t':\mathbb{Q}$ with $A'\cap\{\gamma\mid(\gamma,x)\in d\}=\emptyset$, then $A\cup A';\Gamma\cup\Gamma';\overline{d}\vdash u[t/x]\;\hat{\sim}\;u'[t'/x]:\mathbb{P}$, where $\overline{d}=(d\setminus x)\cup d'\cup\{\{\beta\}\times(A',\Gamma')\mid(\beta,x)\in d\}$;*

2. *if $A,\alpha;\Gamma;d\vdash u\;\hat{\sim}\;u':\mathbb{P}$, and $N=\{\gamma\mid(\alpha,\gamma)\in d\}$, then for all name variables $\beta\in(A\setminus N)$ it holds $A;\Gamma;d[\beta/\alpha]\vdash u[\beta/\alpha]\;\hat{\sim}\;u'[\beta/\alpha]:\mathbb{P}$.*

Proof Both parts are proved by induction on the depth of the derivation respectively of $A;\Gamma,x{:}\mathbb{P};d\vdash u\;\hat{\sim}\;u':\mathbb{P}$ and $A,\alpha;\Gamma;d\vdash u\;\hat{\sim}\;u':\mathbb{P}$.

To illustrate the proof, we focus on part 1, and we detail the case when the last rule of the derivation is new-name abstraction. Suppose that the conclusion of the derivation is $A;\Gamma,x{:}\mathbb{Q};d\vdash \mathit{new}\alpha.u\;\hat{\sim}\;u':\delta\mathbb{P}$. This must have been derived from $A,\beta;\Gamma,x{:}\mathbb{Q};d,(\{\beta\}\times(A\cup(\Gamma,x{:}\mathbb{Q})))\vdash u[\beta/\alpha]\;\hat{\sim}\;v[\beta/\alpha]:\mathbb{P}$ and $A;\Gamma,x{:}\mathbb{Q};d\vdash \mathit{new}\alpha.v\sim^\circ u':\delta\mathbb{P}$ for some term v and for some fresh name β. As α is bound in $\mathit{new}\alpha.u$, we assume without loss of generality that $\alpha\notin A'$. More interestingly, as a consequence of Lemma A.7, we can also

$$\frac{A;\Gamma;d \vdash t \mathrel{\hat{\sim}} w : \mathbb{P}}{A';\Gamma';d' \vdash t \mathrel{\hat{\sim}} w : \mathbb{P}}\ \begin{matrix} A' \supseteq A \\ \Gamma' \supseteq \Gamma \\ d' \supseteq d \end{matrix} \qquad \frac{E \vdash t \mathrel{\hat{\sim}} w : \mathbb{P}_j[\mu\vec{P}.\vec{\mathbb{P}}/\vec{P}]}{E \vdash t \mathrel{\hat{\sim}} w : \mu_j P : \vec{\mathbb{P}}} \qquad \frac{E \vdash \mathbf{0} \sim^\circ w : \mathbb{P}}{E \vdash \mathbf{0} \mathrel{\hat{\sim}} w : \mathbb{P}} \qquad \frac{E \vdash x \sim^\circ w : \mathbb{P}}{E \vdash x \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \qquad E \vdash !t' \sim^\circ w : !\mathbb{P}}{E \vdash !t \mathrel{\hat{\sim}} w : !\mathbb{P}} \qquad \frac{A;\Gamma,x{:}\mathbb{P};d \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \qquad A;\Gamma;d \vdash \mathit{rec}x.t' \sim^\circ w : \mathbb{P}}{A;\Gamma;d \vdash \mathit{rec}x.t \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \qquad E \vdash n \cdot t' \sim^\circ w : \mathbb{N} \otimes \mathbb{P}}{E \vdash n \cdot t \mathrel{\hat{\sim}} w : \mathbb{N} \otimes \mathbb{P}} \qquad \frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{N} \otimes \mathbb{P} \qquad E \vdash \pi_n t' \sim^\circ w : \mathbb{P}}{E \vdash \pi_n t \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{A;\Gamma,x{:}\mathbb{P};d \vdash t \mathrel{\hat{\sim}} t' : \mathbb{Q} \qquad A;\Gamma;d \vdash \lambda x.t' \sim^\circ w : \mathbb{P} \to \mathbb{Q}}{A;\Gamma;d \vdash \lambda x.t \mathrel{\hat{\sim}} w : \mathbb{P} \to \mathbb{Q}}$$

$$\frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \to \mathbb{Q} \qquad E \vdash u \mathrel{\hat{\sim}} u' : \mathbb{P} \qquad E \vdash t'u' \sim^\circ w : \mathbb{P}}{E \vdash tu \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{A,\alpha;\Gamma;d \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \quad A;\Gamma;d \vdash \lambda\alpha.t' \sim^\circ w : \mathbb{N} \to \mathbb{P}}{A;\Gamma;d \vdash \lambda\alpha.t \mathrel{\hat{\sim}} w : \mathbb{N} \to \mathbb{P}} \qquad \frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{N} \to \mathbb{P} \quad E \vdash t'\alpha \sim^\circ w : \mathbb{P}}{E \vdash t\alpha \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{A,\beta;\Gamma;d,(\{\beta\} \times (A,\Gamma)) \vdash t[\beta/\alpha] \mathrel{\hat{\sim}} t'[\beta/\alpha] : \mathbb{P} \qquad A;\Gamma;d \vdash \mathit{new}\alpha.t' \sim^\circ w : \delta\mathbb{P}}{A;\Gamma;d \vdash \mathit{new}\alpha.t \mathrel{\hat{\sim}} w : \delta\mathbb{P}}$$

$$\frac{A;\Gamma;d \vdash t \mathrel{\hat{\sim}} t' : \delta\mathbb{P} \qquad A,\alpha;\Gamma;d \cup (\{\alpha\} \times (A,\Gamma)) \vdash t'[\alpha] \sim^\circ w : \mathbb{P}}{A,\alpha;\Gamma;d \cup (\{\alpha\} \times (A,\Gamma)) \vdash t[\alpha] \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{E \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P}_i \qquad E \vdash i{:}t' \sim^\circ w : \Sigma_i \mathbb{P}_i}{E \vdash i{:}t \mathrel{\hat{\sim}} w : \Sigma_i \mathbb{P}_i} \qquad \frac{E \vdash t \mathrel{\hat{\sim}} t' : \Sigma_{i \in I} \mathbb{P}_i \qquad E \vdash \pi_i t' \sim^\circ w : \mathbb{P}_i}{E \vdash \pi_i t \mathrel{\hat{\sim}} w : \mathbb{P}_i}$$

$$\frac{\forall i \quad E \vdash t_i \mathrel{\hat{\sim}} t'_i : \mathbb{P} \quad E \vdash \Sigma_{i \in I} t'_i \sim^\circ w : \mathbb{P}}{E \vdash \Sigma_{i \in I} t_i \mathrel{\hat{\sim}} w : \mathbb{P}} \qquad \frac{A,\alpha;\Gamma;d \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \quad A;\Gamma;d \vdash \Sigma_{\alpha \in \mathbb{N}} t' \sim^\circ w : \mathbb{P}}{A;\Gamma;d \vdash \Sigma_{\alpha \in \mathbb{N}} t \mathrel{\hat{\sim}} w : \mathbb{P}}$$

$$\frac{A';\Gamma';d' \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \quad A;\Gamma,x{:}\mathbb{R};d \vdash u \mathrel{\hat{\sim}} u' : \mathbb{Q} \quad A \cup A';\Gamma \cup \Gamma';\overline{d} \vdash [t' > p(x) \Rightarrow u'] \sim^\circ w : \mathbb{Q}}{A \cup A';\Gamma \cup \Gamma';\overline{d} \vdash [t > p(x) \Rightarrow u] \mathrel{\hat{\sim}} w : \mathbb{Q}}$$

In the last rule, $\overline{d}$ stands for $(d \setminus x) \cup d' \cup \{\{\alpha\} \times (A',\Gamma') \mid (\alpha,x) \in d\}$. Also, the pattern p is supposed to be well-type, with resumption type $\mathbb{R}$.
In the rule for new-name abstraction, we assume $A,\alpha;\Gamma;d' \vdash t : \mathbb{P}$, with $d = d' \setminus \alpha$. In particular, this implies that $\alpha \notin A$.

Figure 1: The precongruence candidate

suppose $\beta \notin A'$. By the induction hypothesis we get $(A,\beta)\cup A';\Gamma\cup\Gamma';d_1 \vdash u[\beta/\alpha][t/x] \hat{\sim} v[\beta/\alpha][t'/x] : \mathbb{P}$, where $d_1 = ((d,(\{\beta\}\times(A\cup(\Gamma,x{:}\mathbb{Q})))) \setminus x) \cup d' \cup \{\{\alpha\}\times(A'\cup\Gamma') \mid (\alpha,x)\in(d,(\beta\times(A\cup(\Gamma,x{:}\mathbb{Q}))))\}$. The set of distinctions d' can be rearranged so that it is of the form $d_1 = d_2,(\{\beta\}\times(A\cup A'\cup\Gamma\cup\Gamma'))$ for a set of distinctions d_2. Then we have $(A\cup A'),\beta;\Gamma\cup\Gamma';d_1 \vdash u[t/x][\beta/\alpha] \hat{\sim} v[t'/x][\beta/\alpha] : \mathbb{P}$. Since $\sim^\circ$ is defined as the open extension of $\sim$, and because of Lemma A.5, it holds $A\cup A';\Gamma\cup\Gamma';d_2 \vdash (new\alpha.v)[t/x] \sim^\circ u'[t'/x] : \delta\mathbb{P}$ and hence $A;\Gamma;\overline{d} \vdash new\alpha.(v[t/x]) \sim^\circ u'[t'/x] : \delta\mathbb{P}$, where $\overline{d} = d_2 \setminus \alpha$. We conclude $A\cup A';\Gamma\cup\Gamma';\overline{d} \vdash (new\alpha.u)[t/x] \hat{\sim} u'[t'/x] : \delta\mathbb{P}$. □

Part 2 of Lemma A.8 (closure under name substitutions) allows us to prove some basic properties of the precongruence candidate. The proof of these properties involves the closure under name substitution to deal with the rule of new-name abstraction: we will detail one case to illustrate the proof strategy.

Lemma A.9 *Some properties of the precongruence candidate:*

1. $\hat{\sim}$ *is reflexive;*
2. $\hat{\sim}$ *is operator respecting;*
3. $\sim^\circ \subseteq \hat{\sim}$*;*
4. *if* $E \vdash t \hat{\sim} u : \mathbb{P}$ *and* $E \vdash u \sim^\circ v : \mathbb{P}$ *then* $E \vdash t \hat{\sim} v : \mathbb{P}$.

Proof

1. follows from reflexivity of $\sim^\circ$ (by induction on the structure of the term t).
2. follows from the definition of $\hat{\sim}$, from the definition of operator respecting relation, and from reflexivity of $\sim^\circ$. The case of new-name abstraction deserves to be detailed. Suppose $A,\alpha;\Gamma;d \vdash t \hat{\sim} u : \mathbb{P}$. We want to conclude that $A;\Gamma;(d\setminus\alpha) \vdash new\alpha.t \hat{\sim} new\alpha.u : \delta\mathbb{P}$. As $A,\alpha;\Gamma;d \vdash t \hat{\sim} u : \mathbb{P}$, by Lemma A.5 we have $A,\alpha,\beta;\Gamma;d,(\{\beta\}\times(A\cup\Gamma)) \vdash t \hat{\sim} u : \mathbb{P}$. Then, by Lemma A.8 we have $A,\beta;\Gamma;(d,(\{\beta\}\times(A\cup\Gamma)))[\beta/\alpha] \vdash t[\beta/\alpha] \hat{\sim} u[\beta/\alpha] : \mathbb{P}$. Now, $(d\cup(\{\beta\}\times(A\cup\Gamma)))[\beta/\alpha] = (d\setminus\alpha)\cup(\{\beta\}\times(A\cup\Gamma))$. As $\sim^\circ$ is reflexive, $A;\Gamma;(d\setminus\alpha) \vdash new\alpha.u \sim^\circ new\alpha.u : \delta\mathbb{P}$. Hence $A;\Gamma;(d\setminus\alpha) \vdash new\alpha.t \hat{\sim} new\alpha.u : \delta\mathbb{P}$ follows from the definition of $\hat{\sim}$.
3. follows from the reflexivity of $\hat{\sim}$ and the definition of $\hat{\sim}$.
4. induction on the derivation of $E \vdash t \hat{\sim} u$, using the fact that $\sim$ (and $\sim^\circ$) is transitive. □

Lemma A.10 *If* $s \vdash t \hat{\sim}_c u : \mathbb{P}$*, then for all* $s' \supseteq s$ *we have* $s' \vdash t \hat{\sim}_c u : \mathbb{P}$.

Proof Consequence of the weakening rule in the definition of the precongruence candidate. □

Proposition A.11 *Since $\sim$ is an equivalence relation, the transitive closure $\hat{\sim}^*$ of $\hat{\sim}$ is symmetric, and therefore so is $\hat{\sim}_c{}^*$.*

In the next lemma, we heavily rely on the correspondence between the type judgement $s \vdash t : \mathbb{P}$ and the judgement $A; \emptyset; A \vdash \sigma t : \mathbb{P}$ for A a set of fresh name variables and $\sigma : s \to A$ a bijection between s and A.

Lemma A.12 *$\hat{\sim}_c$ is a simulation.*

Proof We prove that $\hat{\sim}_c$ is a simulation by induction on the derivations of the operational semantics. Actually, we prove a stronger property:

> if $s \vdash t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ and $s' \vdash t \xrightarrow{p} t'$ for some $s' \supseteq s$, then for all p' with $s';; x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} p' : \mathbb{P}$, there exists a term u' such that $s' \vdash u \xrightarrow{p'} u'$ and $s' \vdash t' \mathrel{\hat{\sim}_c} u'$.

Since $\hat{\sim}$ is reflexive, $s';; x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} p : \mathbb{P}$ for all actions, and so $\hat{\sim}_c$ is a simulation if the above holds. This stronger induction hypothesis is needed in the case of process application.

Most of the cases are proved in the same way. Consider $E \vdash t : \mathbb{P}$ and $s \vdash C(t) \sim_c u : \mathbb{Q}$ for some term constructor C, possibly involving binding. From the definition of $\hat{\sim}$ we obtain the existence of a term v with $E \vdash t \mathrel{\hat{\sim}} v : \mathbb{P}$ and $s \vdash C(v) \sim u : \mathbb{Q}$. Under the assumption $s' \vdash C(t) \xrightarrow{p(x)} t'$ with $s' \supseteq s$ and with q any action such that $s';; x{:}\mathbb{R} \vdash p \sim q : \mathbb{Q}$, we show that there is a transition $s' \vdash C(v) \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$. Having showed this, *in all cases* we conclude as follows: since $s \vdash C(v) \sim u : \mathbb{Q}$, there is a transition $s' \vdash u \xrightarrow{q(x)} u'$ with $s' \vdash v' \sim u'$. Hence $s' \vdash t' \mathrel{\hat{\sim}_c} u' : \mathbb{R}$ follows from $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$ by Lemma A.9.4. To avoid repetition, this latter part will be left out below.

Cases *sum*, *process application*, and *pattern matching* differ from the above pattern because the constructor C takes more than one term: apart from this, their proof follows the aforementioned pattern.

Prototypical action (prefixing). Suppose $s \vdash !t \mathrel{\hat{\sim}_c} u : !\mathbb{P}$, and $s' \vdash !t \xrightarrow{!(x)} t$ for $s' \supseteq s$. Since $s \vdash !t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{P}$ and $s \vdash !v \sim u : !\mathbb{P}$. By Corollary A.10 we have $s' \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{P}$. We get a transition $s' \vdash !v \xrightarrow{!(x)} v$ from the operational rules.

Tensor. Suppose $s \vdash a \cdot t \mathrel{\hat{\sim}_c} u : \mathbb{N} \otimes \mathbb{P}$ and that $s' \vdash a \cdot t \xrightarrow{a \cdot p(x)} t'$ because $s' \vdash t \xrightarrow{p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';; x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$. This implies $s';; x{:}\mathbb{R} \vdash a \cdot p \mathrel{\hat{\sim}} a \cdot q : \mathbb{N} \otimes \mathbb{P}$. Since $s \vdash a \cdot t \mathrel{\hat{\sim}_c} u : \mathbb{N} \otimes \mathbb{P}$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{P}$ and $s \vdash a \cdot v \sim u : \mathbb{N} \otimes \mathbb{P}$. By the induction hypothesis we get $s' \vdash v \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash a \cdot v \xrightarrow{a \cdot q(x)} v'$.

Projection over names. Suppose $s \vdash \pi_a t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ and that $s' \vdash \pi_a t \xrightarrow{p(x)} t'$ because $s' \vdash t \xrightarrow{a \cdot p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';; x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$. This implies $s';; x{:}\mathbb{R} \vdash a \cdot p \mathrel{\hat{\sim}} a \cdot q : \mathbb{N} \otimes \mathbb{P}$. Since $s \vdash \pi_a t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ there exists a term v

such that $s \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{N} \otimes \mathbb{P}$ and $s \vdash \pi_a v \sim u : \mathbb{P}$. By the induction hypothesis we get $s' \vdash v \xrightarrow{a \cdot q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash \pi_a v \xrightarrow{q(x)} v'$.

Sum. Suppose $s \vdash \Sigma_{i \in I} t_i \mathrel{\hat{\sim}_c} u : \mathbb{P}$ and that $s' \vdash \Sigma_{i \in I} t_i \xrightarrow{p(x)} t'$ is derived from $s' \vdash t_i \xrightarrow{p(x)} t'$, for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$. Since $s \vdash \Sigma_{i \in I} t_i \mathrel{\hat{\sim}_c} u : \mathbb{P}$, there is a family of terms $\{v_i\}_{i \in I}$ such that $s \vdash t_i \mathrel{\hat{\sim}_c} v_i : \mathbb{P}$ for each $i \in I$, and $s \vdash \Sigma_{i \in I} v_i \sim u : \mathbb{P}$. By the induction hypothesis we get $s' \vdash v_i \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash \Sigma_{i \in I} v_i \xrightarrow{q(x)} v'$.

Sum over names. Suppose $s \vdash \Sigma_{\alpha \in \mathbb{N}} t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ and that $s' \vdash \Sigma_{\alpha \in \mathbb{N}} t \xrightarrow{p(x)} t'$ is derived from $s' \vdash t[a/\alpha] \xrightarrow{p(x)} t'$ where $a \in s'$, for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$. Since $s \vdash \Sigma_{\alpha \in \mathbb{N}} t \mathrel{\hat{\sim}_c} u : \mathbb{P}$, there exists a term v such that $A, \alpha; \emptyset; A \vdash \sigma t \mathrel{\hat{\sim}} \sigma v : \mathbb{P}$ for a bijection $\sigma : s \to A$, and $s \vdash \Sigma_{\alpha \in \mathbb{N}} v \sim u : \mathbb{P}$. Using weakening and Lemma A.8.2, we obtain $s' \vdash t[a/\alpha] \mathrel{\hat{\sim}_c} v[a/\alpha]$. By the induction hypothesis we get $s' \vdash v[a/\alpha] \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash \Sigma_{\alpha \in \mathbb{N}} v \xrightarrow{q(x)} v'$.

Recursion. Suppose $s \vdash \mathit{rec}\, y.t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ and that $s' \vdash \mathit{rec}\, y.t \xrightarrow{p(x)} t'$ is derived from $s' \vdash t[\mathit{rec}\, y.t/y] \xrightarrow{p(x)} t'$. Let q be any action with $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$. Since $s \vdash \mathit{rec}\, y.t \mathrel{\hat{\sim}_c} u : \mathbb{P}$ there exists a term v such that $A; y{:}\mathbb{P}; A \vdash \sigma t \mathrel{\hat{\sim}} \sigma v : \mathbb{Q}$ for a bijection $\sigma : s \to A$, and $s \vdash \mathit{rec}\, y.v \sim u : \mathbb{P} \to \mathbb{Q}$. As $\hat{\sim}$ is operator respecting, we have $s \vdash \mathit{rec}\, y.t \mathrel{\hat{\sim}_c} \mathit{rec}\, y.v : \mathbb{P}$, and using Lemma A.8.1 we obtain $s \vdash t[\mathit{rec}\, y.t/y] \mathrel{\hat{\sim}_c} v[\mathit{rec}\, y.v/y] : \mathbb{P}$. By the induction hypothesis we get $s' \vdash v[\mathit{rec}\, y.v/y] \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence also $s' \vdash \mathit{rec}\, y.v \xrightarrow{q(x)} v'$.

Process abstraction. Suppose $s \vdash \lambda y.t \mathrel{\hat{\sim}_c} u : \mathbb{P} \to \mathbb{Q}$ and that $s' \vdash \lambda y.t \xrightarrow{w_1 \mapsto p(x)} t'$ is derived from $s' \vdash t[w_1/y] \xrightarrow{p(x)} t'$. Let $w_2 \mapsto q$ be any action with $s';;x{:}\mathbb{R} \vdash w_1 \mapsto p \mathrel{\hat{\sim}} w_2 \mapsto q : \mathbb{P} \to \mathbb{Q}$. This implies $s' \vdash w_1 \mathrel{\hat{\sim}_c} w_2 : \mathbb{P}$. Since $s \vdash \lambda y.t \mathrel{\hat{\sim}_c} u : \mathbb{P} \to \mathbb{Q}$ there exists a term v such that $A; y{:}\mathbb{P}; A \vdash \sigma t \mathrel{\hat{\sim}} \sigma v : \mathbb{Q}$ for $\sigma : s \to A$, and $s \vdash \lambda y.v \sim u : \mathbb{P} \to \mathbb{Q}$. Using weakening and Lemma A.8.1 we have $s' \vdash t[w_1/y] \mathrel{\hat{\sim}_c} v[w_2/y] : \mathbb{Q}$. By the induction hypothesis we get $s' \vdash v[w_2/y] \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence also $s' \vdash \lambda y.v \xrightarrow{w_2 \mapsto q(x)} v'$.

Process application. Suppose $s \vdash t_1 t_2 \mathrel{\hat{\sim}_c} u : \mathbb{Q}$ and that $s' \vdash t_1 t_2 \xrightarrow{p(x)} t'$ is derived from $s' \vdash t_1 \xrightarrow{t_2 \mapsto p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{Q}$. Since $s \vdash t_1 t_2 \mathrel{\hat{\sim}_c} u : \mathbb{Q}$ there exists terms v_1 and v_2 such that $s \vdash t_1 \mathrel{\hat{\sim}_c} v_1 : \mathbb{P} \to \mathbb{Q}$ and $s \vdash t_2 \mathrel{\hat{\sim}_c} v_2 : \mathbb{P}$ and $s \vdash v_1 v_2 \sim u : \mathbb{Q}$. By the induction hypothesis we get $s' \vdash v_1 \xrightarrow{v_2 \mapsto q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash v_1 v_2 \xrightarrow{v_2 \mapsto q(x)} v'$. Notice how the stronger induction hypothesis allows us to choose the label $v_2 \mapsto q$ rather than $u_2 \mapsto p$, so that we could obtain a transition from $v_1 v_2$.

Name abstraction. Suppose $s \vdash \lambda \alpha.t \mathrel{\hat{\sim}_c} u : \mathbb{N} \to \mathbb{Q}$ and that $s' \vdash \lambda \alpha.t \xrightarrow{a \mapsto p(x)} t'$ is derived from $s' \vdash t[a/\alpha] \xrightarrow{p(x)} t'$. Let $a \mapsto q$ be any action with $s';;x{:}\mathbb{R} \vdash a \mapsto p \mathrel{\hat{\sim}} a \mapsto q : \mathbb{N} \to \mathbb{Q}$. Since $s \vdash \lambda \alpha.t \mathrel{\hat{\sim}_c} u : \mathbb{N} \to \mathbb{Q}$ there exists a term v such that $A, \alpha; \emptyset; A \vdash \sigma t \mathrel{\hat{\sim}} \sigma v : \mathbb{Q}$ for a bijection $\sigma : s \to A$, and $s \vdash \lambda \alpha.v \sim u : \mathbb{N} \to \mathbb{Q}$. Using weakening and Lemma A.8.2 we have $s' \vdash t[a/\alpha] \mathrel{\hat{\sim}_c} v[a/\alpha] : \mathbb{Q}$. By the induction hypothesis we get $s' \vdash v[a/\alpha] \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence also $s' \vdash \lambda\alpha.v \xrightarrow{a \mapsto q(x)} v'$.

Name application. Suppose $s \vdash ta \mathrel{\hat{\sim}_c} u : \mathbb{Q}$ and that $s' \vdash ta \xrightarrow{p(x)} t'$ is derived from $s' \vdash t \xrightarrow{a \mapsto p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{Q}$. Since $s \vdash ta \mathrel{\hat{\sim}_c} u : \mathbb{Q}$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{N} \to \mathbb{Q}$ and $s \vdash va \sim u : \mathbb{Q}$. By the induction hypothesis we get $s' \vdash v \xrightarrow{a \mapsto q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash va \xrightarrow{a \mapsto q(x)} v'$.

Injection. Suppose $s \vdash i{:}t \mathrel{\hat{\sim}_c} u : \Sigma_{i \in I}\mathbb{P}_i$ and that $s' \vdash i{:}t \xrightarrow{i:p(x)} t'$ because $s' \vdash t \xrightarrow{p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}_i$. This implies $s';;x{:}\mathbb{R} \vdash i{:}p \mathrel{\hat{\sim}} i{:}q : \Sigma_{i \in I}\mathbb{P}_i$. Since $s \vdash i{:}t \mathrel{\hat{\sim}_c} u : \Sigma_{i \in i}\mathbb{P}_i$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}_c} v : \mathbb{P}_i$ and $s \vdash i{:}v \sim u : \Sigma_{i \in I}\mathbb{P}_i$. By the induction hypothesis we get $s' \vdash v \xrightarrow{q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash a \cdot v \xrightarrow{i:q(x)} v'$.

Projection. Suppose $s \vdash \pi_i t \mathrel{\hat{\sim}_c} u : \mathbb{P}_i$ and that $s' \vdash \pi_i t \xrightarrow{p(x)} t'$ because $s' \vdash t \xrightarrow{i:p(x)} t'$ for some $s' \supseteq s$. Let q be any action such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}_i$. This implies $s';;x{:}\mathbb{R} \vdash i{:}p \mathrel{\hat{\sim}} i{:}q : \Sigma_{i \in I}\mathbb{P}_i$. Since $s \vdash \pi_i t \mathrel{\hat{\sim}_c} u : \mathbb{P}_i$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}_c} v : \Sigma_{i \in I}\mathbb{P}_i$ and $s \vdash \pi_i v \sim u : \mathbb{P}_i$. By the induction hypothesis we get $s' \vdash v \xrightarrow{i:q(x)} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \mathbb{R}$, and hence $s' \vdash \pi_i v \xrightarrow{q(x)} v'$.

New-name abstraction. Suppose $s \vdash \mathit{new}\alpha.t \mathrel{\hat{\sim}_c} u : \delta\mathbb{P}$. Suppose also that $s' \vdash \mathit{new}\alpha.t \xrightarrow{\mathit{new}\alpha.p[x'[\alpha]/x](x')} \mathit{new}\alpha.t'$ is derived from $s' \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha](x)} t'[a/\alpha]$ for some $s' \supseteq s$. Let q be any action such that $A', \alpha; \emptyset; A', d;;x : \mathbb{R} \vdash \sigma p \mathrel{\hat{\sim}} \sigma q : \mathbb{P}$ for a bijection $\sigma : s' \to A'$. This implies $s' \mathbin{\dot\cup} \{a\};;x{:}\mathbb{R} \vdash p[a/\alpha] \mathrel{\hat{\sim}} q[a/\alpha] : \mathbb{P}$ and $s';;x'{:}\delta\mathbb{R} \vdash \mathit{new}\alpha.p \mathrel{\hat{\sim}} \mathit{new}\alpha.q : \delta\mathbb{P}$. Since $s \vdash \mathit{new}\alpha.t \mathrel{\hat{\sim}_c} u : \delta\mathbb{P}$ there exists a term v such that $s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \mathrel{\hat{\sim}_c} v[a/\alpha] : \mathbb{P}$ and $s \vdash \mathit{new}\alpha.v \sim u : \delta\mathbb{P}$. As $s' \cup \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha](x)} t'[a/\alpha]$, by the induction hypothesis we have $s' \cup \{a\} \vdash v[a/\alpha] \xrightarrow{q[a/\alpha](x)} v'[a/\alpha]$ with $s' \mathbin{\dot\cup} \{a\} \vdash t'[a/\alpha] \mathrel{\hat{\sim}_c} v'[a/\alpha] : \mathbb{P}$. By the operational rules $s' \vdash \mathit{new}\alpha.v \xrightarrow{\mathit{new}\alpha.q[x'[\alpha]/x](x')} \mathit{new}\alpha.v'$. As $\sim$ is reflexive, we can deduce $s' \vdash \mathit{new}\alpha.t' \mathrel{\hat{\sim}_c} \mathit{new}\alpha.v' : \delta\mathbb{P}$ from $s' \mathbin{\dot\cup} \{a\} \vdash t'[a/\alpha] \mathrel{\hat{\sim}_c} v'[a/\alpha] : \mathbb{P}$, as desired.

New name application. Suppose $s \mathbin{\dot\cup} \{a\} \vdash t[a] \mathrel{\hat{\sim}_c} u : \mathbb{P}$. Suppose also that $s' \mathbin{\dot\cup} \{a\} \vdash t[a] \xrightarrow{p[a/\alpha](x)} t'[a]$ is derived from $s' \vdash t \xrightarrow{\mathit{new}\alpha.p[x'[\alpha]/x](x')} t'$, for some $s' \supseteq s$. Let q be any action such that $A', \alpha; \emptyset; A', d;;x : \mathbb{R} \vdash \sigma p \mathrel{\hat{\sim}} \sigma q : \mathbb{P}$ for a bijection $\sigma : s' \to A'$. This implies $s';;x' : \delta\mathbb{R} \vdash \mathit{new}\alpha.p[x'[\alpha]/x] \mathrel{\hat{\sim}} \mathit{new}\alpha.q[x'[\alpha]/x] : \delta\mathbb{P}$, and also $s' \mathbin{\dot\cup} \{a\};;x : \mathbb{R} \vdash p[a/\alpha] \mathrel{\hat{\sim}} q[a/\alpha] : \mathbb{P}$. Since $s \mathbin{\dot\cup} \{a\} \vdash t[a] \mathrel{\hat{\sim}_c} u : \mathbb{P}$ there exists a term v such that $s \vdash t \mathrel{\hat{\sim}} v : \delta\mathbb{P}$ and $s \mathbin{\dot\cup} \{a\} \vdash v[a] \sim u : \mathbb{P}$. By the induction hypothesis we have $s' \vdash v \xrightarrow{\mathit{new}\alpha.q[x'[\alpha]/x]} v'$ with $s' \vdash t' \mathrel{\hat{\sim}_c} v' : \delta\mathbb{R}$. As $\hat{\sim}$ is operator respecting (Lemma A.9.2), we obtain $s' \mathbin{\dot\cup} \{a\} \vdash t'[a] \mathrel{\hat{\sim}_c} v'[a] : \mathbb{P}$. We get a transition $s' \mathbin{\dot\cup} \{a\} \vdash v[a] \xrightarrow{q[a/\alpha]} v'[a]$ by the operational rules.

Pattern matching. Suppose $s \vdash [t_1 > p(x) \Rightarrow t_2] \mathrel{\hat{\sim}_c} u : \mathbb{Q}$ and that $s' \vdash [t_1 > p(x) \Rightarrow t_2] \xrightarrow{p'(x')} t_3$ is derived from $s' \vdash t_1 \xrightarrow{p(x)} t'_1$ and $s' \vdash t_2[t'_1/x] \xrightarrow{p'(x')} t_3$, for some $s' \supseteq s$. Let q and q' be actions such that $s';;x{:}\mathbb{R} \vdash p \mathrel{\hat{\sim}} q : \mathbb{P}$ and $s';;x'{:}\mathbb{R}' \vdash p' \mathrel{\hat{\sim}_c} q' : \mathbb{Q}$. Since

$s \vdash [t_1 > p(x) \Rightarrow t_2] \mathrel{\hat{\sim}_c} u : \mathbb{Q}$, there exist two terms v_1 and v_2 such that $s_1 \vdash t_1 \mathrel{\hat{\sim}_c} v_1 : \mathbb{P}$ and $A; x{:}\mathbb{R}; A, d \vdash \sigma t_2 \mathrel{\hat{\sim}_c} \sigma v_2 : \mathbb{Q}$ for a bijection $\sigma : s_2 \to A$, where $s_1 \cup s_2 = s$ and $s' \cap \sigma(\{\alpha \mid (\alpha, x) \in d\}) = \emptyset$. It also holds $s \vdash [v_1 > p(x) \Rightarrow v_2] \sim u : \mathbb{Q}$. By the induction hypothesis we have $s' \vdash v_1 \xrightarrow{q(x)} v_1'$ with $s' \vdash t_1' \mathrel{\hat{\sim}_c} v_1' : \mathbb{R}$. By weakening and by Lemma A.8.1 we get $s' \vdash t_2[t_1'/x] \mathrel{\hat{\sim}_c} v_2[v_1'/x] : \mathbb{Q}$. Applying the induction hypothesis in this case yields $s' \vdash v_2[v_1'/x] \xrightarrow{q'(x')} v_3$ with $s' \vdash t_3 \mathrel{\hat{\sim}_c} v_3$, and hence $s' \vdash [v_1 > p(x) \Rightarrow v_2] \xrightarrow{q'(x')} v_3$.

The induction is complete. $\square$

At last, we prove that bisimilarity $\sim$ is a congruence.

Proof of Theorem 3.2, page 12 As shown in Lemma A.12, $\hat{\sim}_c$ is a simulation. Then $\hat{\sim}_c{}^*$ is a bisimulation by Property A.11, and so $\hat{\sim}_c{}^* \subseteq \sim$. In particular $\hat{\sim}_c \subseteq \sim$. By Lemma A.9.1 and Lemma A.8, it follows that $\hat{\sim} \subseteq \sim^\circ$, and so by Lemma A.9.3, $\hat{\sim} = \sim^\circ$. Hence, $\sim$ is a congruence because it is an equivalence relation and by Lemma A.9.2 is operator respecting. $\square$

A remark on the definition of the precongruence candidate: the more standard rule

$$\frac{A, \alpha; \Gamma; d \vdash t \mathrel{\hat{\sim}} t' : \mathbb{P} \qquad A; \Gamma; (d \setminus \alpha) \vdash new\alpha.t' \sim^\circ w : \delta\mathbb{P}}{A; \Gamma; (d \setminus \alpha) \vdash new\alpha.t \mathrel{\hat{\sim}} w : \delta\mathbb{P}}$$

does not seem to capture the essence of new-name abstraction. In fact, this rule does not allow to prove Lemma A.12 (at least not in a handy way).

Proof of Proposition 3.3, page 12 Let $\mathcal{I}$ be the identical type respecting relation over closed terms. In each postulated case $s \vdash lhs \sim rhs : \mathbb{P}$, the relation $\mathcal{S} = \{ (s' \vdash lhs : \mathbb{P}, s' \vdash rhs : \mathbb{P}) \mid s' \supseteq s \} \cup \mathcal{I}$ is a bisimulation. $\square$

Proof of Lemma 3.5, page 12 Let

$$\mathcal{R} = \{ (s \vdash new\alpha.t : \delta\mathbb{P}, s \vdash new\alpha.u : \delta\mathbb{P}) \mid s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \sim u[a/\alpha] : \mathbb{P} \} .$$

We show that $\mathcal{R}$ is a bisimulation. Suppose $s' \vdash new\alpha.t \xrightarrow{new\alpha.p} new\alpha.t'$ for some $s' \supseteq s$. This must have been derived from $s' \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha]} t'[a/\alpha]$ for some a. By bisimulation, we have $s' \mathbin{\dot\cup} \{a\} \vdash u[a/\alpha] \xrightarrow{p[a/\alpha]} u'[a/\alpha]$ with $s' \mathbin{\dot\cup} \{a\} \vdash t'[a/\alpha] \sim u'[a/\alpha] : \mathbb{P}$. By the operational rules, we have $s' \vdash new\alpha.u \xrightarrow{new\alpha.p} new\alpha.u'$, and by definition of $\mathcal{R}$ we conclude $s \vdash new\alpha.t' \sim new\alpha.u' : \delta\mathbb{P}$. $\square$

Proof of Proposition 3.6, page 12 Let

$$\begin{aligned}\mathcal{R} = \{\ & (s \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] : \mathbb{P}, s \mathbin{\dot\cup} \{a\} \vdash t[a/\alpha] : \mathbb{P}) \mid \\ & \qquad A, \alpha; \emptyset; A, d \vdash t : \mathbb{P} \text{ for } \sigma : s \to_{\mathrm{bij}} A\ \} .\end{aligned}$$

We show that $\mathcal{R}$ is a bisimulation. Consider $s \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] \mathrel{\mathcal{R}} t[a/\alpha] : \mathbb{P}$.

Suppose that $s' \mathbin{\dot\cup} \{a\} \vdash (new\alpha.t)[a] \xrightarrow{p[a/\alpha](x)} (new\alpha.t')[a]$ for some $s' \supseteq s$. This must have been derived from $s' \vdash new\alpha.t \xrightarrow{new\alpha.\alpha \mapsto p[x'[\alpha]/x](x')} new\alpha.t'$. In turn, this must

Actions: $\ell \ ::= \ \overline{n}m \mid \overline{n}(\alpha) \mid n\alpha \mid \tau$

$$\frac{}{\overline{n}m.P \xrightarrow{\overline{n}m}_l P} \qquad \frac{}{n(\alpha) \xrightarrow{n(\alpha)}_l P} \qquad \frac{P \xrightarrow{\ell}_l P' \quad \alpha \notin \mathrm{fv}(\ell)}{(\boldsymbol{\nu}\alpha)P \xrightarrow{\ell}_l (\boldsymbol{\nu}\alpha)P'} \qquad \frac{P \xrightarrow{\overline{n}\alpha}_l P'}{(\boldsymbol{\nu}\alpha)P \xrightarrow{\overline{n}(\alpha)}_l P'}$$

$$\frac{P \xrightarrow{\ell}_l P'}{P \mid Q \xrightarrow{\ell}_l P' \mid Q} \qquad \frac{P \xrightarrow{\overline{n}m}_l P' \quad Q \xrightarrow{n\alpha}_l Q'}{P \mid Q \xrightarrow{\tau}_l P' \mid Q'[m/\alpha]} \qquad \frac{P \xrightarrow{\overline{n}(\beta)}_l P' \quad Q \xrightarrow{n\alpha}_l Q'}{P \mid Q \xrightarrow{\tau}_l (\boldsymbol{\nu}\beta)(P' \mid Q'[\beta/\alpha])}$$

Figure 2: π-calculus: the *late* labelled transition system

have derived from $s' \dot\cup \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha](x)} t'[a/\alpha]$. So we have a matching transition, and $s' \dot\cup \{a\} \vdash (new\alpha.t')[a] \ \mathcal{R}\ t'[a/\alpha] : \mathbb{R}$ follows from the construction of $\mathcal{R}$, where $\mathbb{R}$ is the type of the resumption variable in p.

Suppose now that $s' \dot\cup \{a\} \vdash t[a/\alpha] \xrightarrow{p[a/\alpha](x)} t'[a/\alpha]$ for some $s' \supseteq s$. By the operational rules we get $s' \vdash new\alpha.t \xrightarrow{new\alpha.\alpha \mapsto p[x'[\alpha]/x](x')} new\alpha.t'$. In turn, by the operational rules we get $s \dot\cup \{a\} \vdash (new\alpha.t)[a] \xrightarrow{p[a/\alpha](y)} (new\alpha.t')[a]$. So we have a matching transition, and $s' \dot\cup \{a\} \vdash (new\alpha.t')[a] \ \mathcal{R}\ t'[a/\alpha] : \mathbb{R}$ follows from the construction of $\mathcal{R}$. $\Box$

Proof of Corollary 3.7, page 12 By weakening, $s \dot\cup \{a\} \vdash new\alpha.t \sim new\alpha.u : \delta\mathbb{P}$. By congruence, $s \dot\cup \{a\} \vdash (new\alpha.t)[a] \sim new(\alpha.u)[a] : \mathbb{P}$. By Proposition 3.6, we have $s \dot\cup \{a\} \vdash (new\alpha.t)[a/\alpha] \sim t[a/\alpha] : \mathbb{P}$ and $s \dot\cup \{a\} \vdash (new\alpha.u)[a/\alpha] \sim u[a/\alpha] : \mathbb{P}$. The result follows from transitivity of $\sim$. $\Box$

Proof of Corollary 3.8, page 12 By Proposition 3.5 $s \vdash new\alpha.t \sim new\alpha.u : \delta\mathbb{P}$. The result follows by Corollary 3.7. $\Box$

A.3 Proofs from Section 4

We first introduce a basic up-to proof technique.

Definition A.13 (Bisimulation up to bisimilarity) *A symmetric type respecting relation on closed terms, $\mathcal{R}$, is a* bisimulation up to bisimilarity *if $s \vdash t \ \mathcal{R}\ u : \mathbb{P}$ and $s' \vdash t \xrightarrow{p} t'$ for $s' \supseteq s$ imply that there exists a term u' such that $s' \vdash u \xrightarrow{p} u'$ and $s' \vdash t' \sim\mathcal{R}\sim u' : \mathbb{P}$.*

Proposition A.14 *If $\mathcal{R}$ is a bisimulation up to bisimilarity, then $\mathcal{R} \subseteq \sim$.*

Proof Let $\mathcal{S} = \{(s \vdash t : \mathbb{P}, s \vdash u : \mathbb{P}) \mid s \vdash t \sim\mathcal{R}\sim u : \mathbb{P}\}$. The relation $\mathcal{S}$ is a bisimulation (simple diagram chasing argument). $\Box$

For reference, the late labelled transition system is reported in Figure 2 (we omit the symmetric rules).

Definition A.15 (Late strong bisimilarity) Late strong bisimilarity *is the largest symmetric relation, $\sim_l$, such that whenever $P \sim_l Q$,*

1. $P \xrightarrow{n\beta}_l P'$ *implies there is* Q' *such that* $Q \xrightarrow{n\beta}_l Q'$ *and* $P'[m/\alpha] \sim_l Q'[m/\alpha]$ *for every* m*;*

2. *if* ℓ *is not an input action then* $P \xrightarrow{\ell}_l P'$ *implies* $Q \xrightarrow{\ell}_l \sim_l Q'$*.*

It is well-known that late strong bisimilarity is preserved by all operators except input prefix. In particular, both transitions and late strong bisimilarity are preserved by injective renaming.

Lemma A.16 (Basic properties of $[\![-]\!]$)

1. $\mathrm{fv}(P) = \mathrm{fv}([\![P]\!])$*;*

2. $\mathsf{n}(P) = \mathsf{n}([\![P]\!])$*;*

3. $[\![P]\!][a/\alpha] = [\![P[a/\alpha]]\!]$*.*

To prove Theorem 4.2, we introduce here a theorem stronger than Theorem 4.1.

Theorem A.17 *Let* P *a closed* π*-calculus process. Then,*

1. $P \xrightarrow{\overline{\alpha}\beta}_l P'$ *if and only if* $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\mathit{out}:a\cdot b\cdot !} [\![P']\!]$*;*

2a. $P \xrightarrow{\alpha\beta}_l P'$ *implies* $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\mathit{inp}:a!}\sim (\lambda\beta.[\![P']\!])$*;*

2b. $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\mathit{inp}:a!} t$ *implies* $P \xrightarrow{\alpha\beta}_l P'$ *and* $t \sim (\lambda\beta.[\![P']\!])$*;*

3a. $P \xrightarrow{\overline{\alpha}(\beta)}_l P'$ *implies* $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\mathit{bout}:a!}\sim (\mathit{new}\beta.[\![P']\!])$*;*

3b. $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\mathit{bout}:a!} t$ *implies* $P \xrightarrow{\overline{\alpha}(\beta)}_l P'$ *and* $t \sim (\mathit{new}\beta.[\![P']\!])$*;*

4a. $P \xrightarrow{\tau}_l P'$ *implies* $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\tau:!}\sim [\![P']\!]$*;*

4b. $\mathsf{n}(P) \vdash [\![P]\!] \xrightarrow{\tau:!} t$ *implies* $P \xrightarrow{\tau}_l P'$ *and* $t \sim [\![P']\!]$

We introduce some notations useful in the proofs of the next two theorems:

- we write $\vec{\alpha}_n$ or simply $\vec{\alpha}$ for the set $\{\alpha_1, \ldots, \alpha_n\}$ where the α_i are all distinct. We write $\mathit{new}\vec{\alpha}_n.t$ for $\mathit{new}\alpha_1.\ldots.\mathit{new}\alpha_n.t$. Also $\delta^n\mathbb{P}$ stands for $\delta\ldots\delta\mathbb{P}$ where the δ is replicated n times;

- most of the substitutions we use in the next two theorems are bijections involving fresh name constants. So, whenever a is fresh for P, we write $P[a/\alpha] \xrightarrow{\tau} P'[a/\alpha]$ as a shorthand for $P[a/\alpha] \xrightarrow{\tau} P'_1$ and $P' = P'_1[\alpha/a]$. Same with terms and transitions of new-HOPLA;

- we write $\rightarrow$ instead of $\rightarrow_l$.

We prove separately the two implications of Theorem 4.2.

Theorem A.18 *Let P and Q be two closed π-calculus processes. If $P \sim_l Q$ then* $\mathsf{n}(P,Q) \vdash [\![P]\!] \sim [\![Q]\!] : \mathbb{P}$.

Proof We actually prove a stronger theorem:

Let P and Q be two π-calculus processes such that $\mathrm{fv}(P) = \mathrm{fv}(Q) = \vec{\alpha}_n$.

1. If $P \sim_l Q$, then $\mathsf{n}(\mathit{lhs},\mathit{rhs}) \vdash \mathit{new}\vec{\alpha}_n.[\![P]\!] \sim \mathit{new}\vec{\alpha}_n.[\![Q]\!] : \delta^n\mathbb{P}$, and
2. if $\gamma \in \vec{\alpha}$ and for all m it holds $P[m/\gamma] \sim_l Q[m/\gamma]$, then $\mathsf{n}(\mathit{lhs},\mathit{rhs}) \vdash \mathit{new}(\vec{\alpha}_n \setminus \gamma).\lambda\gamma.[\![P]\!] \sim \mathit{new}(\vec{\alpha}_n \setminus \gamma).\lambda\gamma.[\![Q]\!] : \delta^{n-1}(\mathbb{N} \to \mathbb{P})$.

Let

$$\begin{aligned}\mathcal{R} \;=\; & \{\; (s \vdash \mathit{new}\vec{\alpha}_n.[\![P]\!] : \delta^n\mathbb{P}\,,\; s \vdash \mathit{new}\vec{\alpha}_n.[\![Q]\!] : \delta^n\mathbb{P}) \;\mid\; P \sim_l Q \text{ and } \mathrm{fv}(P) = \mathrm{fv}(Q) = \vec{\alpha} \;\}\\ & \cup \{\; (s \vdash \mathit{new}\vec{\alpha}_n.\lambda\gamma.[\![P]\!] : \delta^n(\mathbb{N} \to \mathbb{P})\,,\; s \vdash \mathit{new}\vec{\alpha}_n.\lambda\gamma.[\![Q]\!] : \delta^n(\mathbb{N} \to \mathbb{P})) \;\mid\\ & \qquad \forall m.P[m/\gamma] \sim_l Q[m/\gamma] \text{ and } \mathrm{fv}(P) = \mathrm{fv}(Q) = \vec{\alpha} \,\dot\cup\, \{\gamma\} \;\}\end{aligned}$$

where $s \supseteq \mathsf{n}(\mathit{lhs},\mathit{rhs})$. We prove that $\mathcal{R}$ is a bisimulation up to bisimulation; the result will follows from the soundness of the up-to bisimulation proof technique (Proposition A.14).

First consider

$$s \vdash \mathit{new}\vec{\alpha}_n.[\![P]\!] \;\mathcal{R}\; \mathit{new}\vec{\alpha}_n.[\![Q]\!] : \delta^n\mathbb{P}$$

with $P \sim_l Q$ and $\mathrm{fv}(P) = \mathrm{fv}(Q) = \vec{\alpha}$. We perform a case analysis on the actions performed by $\mathit{new}\vec{\alpha}_n.[\![P]\!]$.

- Suppose that $s' \vdash \mathit{new}\vec{\alpha}_n.[\![P]\!] \xrightarrow{\mathit{new}\vec{\alpha}_n.\tau:!} \mathit{new}\vec{\alpha}_n.t$ for some $s' \supseteq s$. This must have been derived from $s' \,\dot\cup\, \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\tau:!} t[\vec{a}/\vec{\alpha}]$. As $[\![P]\!][\vec{a}/\vec{\alpha}] = [\![P[\vec{a}/\vec{\alpha}]]\!]$, by Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}] \xrightarrow{\tau} P'[\vec{a}/\vec{\alpha}]$ and $s' \,\dot\cup\, \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim [\![P']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. Late strong bisimulation is preserved by injective substitution, so $P[\vec{a}/\vec{\alpha}] \sim_l Q[\vec{a}/\vec{\alpha}]$. Then, by bisimulation there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}] \xrightarrow{\tau} Q'[\vec{a}/\vec{\alpha}]$ and $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$. By Lemma A.17, $s' \,\dot\cup\, \vec{a} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\tau:!} u[\vec{a}/\vec{\alpha}]$ and $s' \,\dot\cup\, \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim [\![Q']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. By the operational rules we have $s' \vdash \mathit{new}\vec{\alpha}_n.[\![Q]\!] \xrightarrow{\mathit{new}\vec{\alpha}_n.\tau:!} \mathit{new}\vec{\alpha}_n.u$. We must still show $s' \vdash \mathit{new}\vec{\alpha}_n.t \sim\!\mathcal{R}\!\sim \mathit{new}\vec{\alpha}_n.u : \delta^n\mathbb{P}$. By multiple applications of Lemma 3.5 we derive $s' \vdash \mathit{new}\vec{\alpha}.t \sim \mathit{new}\vec{\alpha}.[\![P']\!] : \delta^n\mathbb{P}$ from $s' \,\dot\cup\, \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim [\![P']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. We also derive $s' \vdash \mathit{new}\vec{\alpha}.u \sim \mathit{new}\vec{\alpha}.[\![Q']\!] : \delta^n\mathbb{P}$ from $s' \,\dot\cup\, \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim [\![Q']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. Again, late strong bisimulation is preserved by injective substitution, so $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$ implies $P' \sim_l Q'$. By construction of $\mathcal{R}$, we finally have

  $$s' \vdash \mathit{new}\vec{\alpha}.t \sim \mathit{new}\vec{\alpha}.[\![P']\!] \;\mathcal{R}\; \mathit{new}\vec{\alpha}.[\![Q']\!] \sim \mathit{new}\vec{\alpha}.u : \delta^n\mathbb{P}\;.$$

  as wanted.

- The case $\mathit{new}\vec{\alpha}.\mathsf{out}{:}a \cdot b \cdot !$ is similar to the previous one.

- Suppose that for some $s' \supseteq s$, $s' \vdash \mathit{new}\vec{\alpha}_n.[\![P]\!] \xrightarrow{\mathit{new}\vec{\alpha}_n.\mathsf{bout}:b\cdot!} \mathit{new}\vec{\alpha}_n.t$. This must have been derived from $s' \,\dot\cup\, \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{bout}:i\cdot!} t[\vec{a}/\vec{\alpha}]$, where $i =$

a_i if $b = \alpha_i$, and $i = b$ otherwise. By Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}] \xrightarrow{\bar{i}(\gamma)} P'[\vec{a}/\vec{\alpha}]$ and $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (new\gamma.[\![P']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. Observe that $\gamma \notin \vec{\alpha}$. Now, by late strong bisimulation, there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}] \xrightarrow{\bar{i}(\gamma)} Q'[\vec{a}/\vec{\alpha}]$ and $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$. By Lemma A.17, $s' \mathbin{\dot\cup} \vec{a} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{bout}:i\cdot!} u[\vec{a}/\vec{\alpha}]$ and $s' \mathbin{\dot\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (new\gamma.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. By the operational rules we have $s' \vdash new\vec{\alpha}_n.[\![Q]\!] \xrightarrow{new\vec{\alpha}_n.\mathsf{bout}:b\cdot!} new\vec{\alpha}_n.u$. We must conclude $s' \vdash new\vec{\alpha}_n.t \sim\mathcal{R}\sim new\vec{\alpha}_n.u : \delta^n\delta\mathbb{P}$. Multiple applications of Lemma 3.5 allow us to derive $s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.new\gamma.[\![P']\!] : \delta^n\delta\mathbb{P}$ from $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (new\gamma.[\![P']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. We also derive $s' \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.new\gamma.[\![Q']\!] : \delta^n\delta\mathbb{P}$ from $s' \mathbin{\dot\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (new\gamma.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. Since $P' \sim_l Q'$, by construction of $\mathcal{R}$, we have

$$s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.new\gamma.[\![P']\!] \mathrel{\mathcal{R}} new\vec{\alpha}.new\gamma.[\![Q']\!] \sim new\vec{\alpha}.u : \delta^n\delta\mathbb{P} \ .$$

- Suppose that for some $s' \supseteq s$, $s' \vdash new\vec{\alpha}_n.[\![P]\!] \xrightarrow{new\vec{\alpha}_n.\mathsf{inp}:b\cdot!} new\vec{\alpha}_n.t$. This must have been derived from $s' \mathbin{\dot\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{inp}:i\cdot!} t[\vec{a}/\vec{\alpha}]$, where $i = a_i$ if $b = \alpha_i$, and $i = b$ otherwise. By Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}] \xrightarrow{i\gamma} P'[\vec{a}/\vec{\alpha}]$ and $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (\lambda\gamma.[\![P']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. Now, by late strong bisimulation, there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}] \xrightarrow{i\gamma} Q'[\vec{a}/\vec{\alpha}]$ and for all m, $P'[\vec{a}/\vec{\alpha}][m/\gamma] \sim_l Q'[\vec{a}/\vec{\alpha}][m/\gamma]$. By Lemma A.17, $s' \mathbin{\dot\cup} \vec{a} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{inp}:i\cdot!} u[\vec{a}/\vec{\alpha}]$ and $s' \mathbin{\dot\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (\lambda\gamma.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. By the operational rules we have the transition $s' \vdash new\vec{\alpha}_n.[\![Q]\!] \xrightarrow{new\vec{\alpha}_n.\mathsf{inp}:b\cdot!} new\vec{\alpha}_n.u$. We must conclude $s' \vdash new\vec{\alpha}_n.t \sim\mathcal{R}\sim new\vec{\alpha}_n.u : \delta^n\mathbb{N} \to \mathbb{P}$. By multiple applications of Lemma 3.5 we derive $s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.\lambda\gamma.[\![P']\!] : \delta^n(\mathbb{N} \to \mathbb{P})$ from $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (\lambda\gamma.[\![P']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. We also derive $s' \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.\lambda\gamma.[\![Q']\!] : \delta^n(\mathbb{N} \to \mathbb{P})$ from $s' \mathbin{\dot\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (\lambda\gamma.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. Since $P' \sim_l Q'$, we have

$$s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.\lambda\gamma.[\![P']\!] \mathrel{\mathcal{R}} new\vec{\alpha}.\lambda\gamma.[\![Q']\!] \sim new\vec{\alpha}.u : \delta^n(\mathbb{N} \to \mathbb{P}) \ .$$

Consider now

$$s \vdash new\vec{\alpha}_n.\lambda\gamma.[\![P]\!] \mathrel{\mathcal{R}} new\vec{\alpha}_n.\lambda\gamma.[\![Q]\!] : \delta^n(\mathbb{N} \to \mathbb{P})$$

because for all m, $P[m/\gamma] \sim_l Q[m/\gamma]$. We perform a case analysis on the actions performed by $new\vec{\alpha}_n.\lambda\gamma.[\![P]\!]$.

- Suppose that $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![P]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\tau:!} new\vec{\alpha}_n.t$ for some $s' \supseteq s$. This must have been derived from $s' \mathbin{\dot\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{e\mapsto\tau:!} t[\vec{a}/\vec{\alpha}]$, that in turn must have been derived from $s' \mathbin{\dot\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\tau:!} t[\vec{a}/\vec{\alpha}]$ for $e \in s' \mathbin{\dot\cup} \vec{a}$. Observe that $e = a_i$ if $n = \alpha_i$ for some i, and $e = n$ otherwise. By Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\tau} P'[\vec{a}/\vec{\alpha}]$ and $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim [\![P']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. As for all m it holds $P[m/\gamma] \sim_l Q[m/\gamma]$, by bisimulation there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\tau} Q'[\vec{a}/\vec{\alpha}]$ and $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$. By Lemma A.17, $s' \mathbin{\dot\cup} \vec{a} \vdash$

$[\![Q]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\tau:!} u[\vec{a}/\vec{\alpha}]$ and $s' \dot{\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim [\![Q']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. By the operational rules we have $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![Q]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\tau:!} new\vec{\alpha}_n.u$. We must conclude $s' \vdash new\vec{\alpha}_n t \sim\mathcal{R}\sim new\vec{\alpha}_n u : \delta^n\mathbb{P}$. By multiple applications of Lemma 3.5 we derive $s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.[\![P']\!] : \delta^n\mathbb{P}$ from $s' \dot{\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim [\![P']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. We also derive $s' \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.[\![Q']\!] : \delta^n\mathbb{P}$ from$s' \dot{\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim [\![Q']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. Late strong bisimulation is preserved by injective substitution, so $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$ implies $P' \sim_l Q'$. By construction of $\mathcal{R}$, we finally have

$$s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.[\![P']\!] \;\mathcal{R}\; new\vec{\alpha}.[\![Q']\!] \sim new\vec{\alpha}.u : \delta^n\mathbb{P} \;.$$

as wanted.

- The case $new\vec{\alpha}.\mathsf{out}{:}a \cdot b \cdot !$ is similar to the previous one.

- Suppose that $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![P]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\mathsf{bout}:b\cdot!} new\alpha_n.t$ for some $s' \supseteq s$. This must have been derived from $s' \dot{\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{e\mapsto\mathsf{bout}:i\cdot!} t[\vec{a}/\vec{\alpha}]$, that in turn must have been derived from $s' \dot{\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\mathsf{bout}:i\cdot!} t[\vec{a}/\vec{\alpha}]$ for $e \in s' \dot{\cup} \vec{a}$. Observe that $e = a_i$ if $n = \alpha_i$ and $e = n$ otherwise. Observe also that $i = a_j$ if $b = \alpha_j$ and $i = b$ otherwise. By Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\bar{i}(\zeta)} P'[\vec{a}/\vec{\alpha}]$ and $s' \dot{\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. As for all m it holds $P[m/\gamma] \sim_l Q[m/\gamma]$, by bisimulation there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\bar{i}(\zeta)} Q'[\vec{a}/\vec{\alpha}]$ and $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$. By Lemma A.17, $s' \dot{\cup} \vec{a} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\mathsf{bout}:i\cdot!} u[\vec{a}/\vec{\alpha}]$ and $s' \dot{\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. By the operational rules we have $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![Q]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\mathsf{bout}:b\cdot!} new\vec{\alpha}_n.u$. We must conclude $s' \vdash new\vec{\alpha}_n t \sim\mathcal{R}\sim new\vec{\alpha}_n u : \delta^n\delta\mathbb{P}$. By multiple applications of Lemma 3.5 we derive $s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.new\zeta.[\![P']\!] : \delta^n\delta\mathbb{P}$ from $s' \dot{\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. We also derive $s' \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.new\zeta.[\![Q']\!] : \delta^n\delta\mathbb{P}$ from $s' \dot{\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. Since $P' \sim_l Q'$, by construction of $\mathcal{R}$, we finally have

$$s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.new\zeta.[\![P']\!] \;\mathcal{R}\; new\vec{\alpha}.new\zeta.[\![Q']\!] \sim new\vec{\alpha}.u : \delta^n\delta\mathbb{P} \;.$$

- Suppose that $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![P]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\mathsf{inp}:b\cdot!} new\alpha_n.t$ for some $s' \supseteq s$. This must have been derived from $s' \dot{\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{e\mapsto\mathsf{inp}:i\cdot!} t[\vec{a}/\vec{\alpha}]$, that in turn must have been derived from $s' \dot{\cup} \vec{a} \vdash [\![P]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\mathsf{inp}:i\cdot!} t[\vec{a}/\vec{\alpha}]$ for $e \in s' \dot{\cup} \vec{a}$. Observe that $e = a_i$ if $n = \alpha_i$ and $e = n$ otherwise. Observe also that $i = a_j$ if $b = \alpha_j$ and $i = b$ otherwise. By Lemma A.17, there is a term $P'[\vec{a}/\vec{\alpha}]$ such that $P[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{i\zeta} P'[\vec{a}/\vec{\alpha}]$ and $s' \dot{\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. As for all m it holds $P[m/\gamma] \sim_l Q[m/\gamma]$, by bisimulation there is $Q'[\vec{a}/\vec{\alpha}]$ such that $Q[\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{i\zeta} Q'[\vec{a}/\vec{\alpha}]$ and $P'[\vec{a}/\vec{\alpha}] \sim_l Q'[\vec{a}/\vec{\alpha}]$. By Lemma A.17, $s' \dot{\cup} \vec{a} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}][e/\gamma] \xrightarrow{\mathsf{inp}:i\cdot!} u[\vec{a}/\vec{\alpha}]$ and $s' \dot{\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. By the operational rules we have $s' \vdash new\vec{\alpha}_n.\lambda\gamma.[\![Q]\!] \xrightarrow{new\vec{\alpha}_n.n\mapsto\mathsf{inp}:b\cdot!} new\vec{\alpha}_n.u$. We must conclude $s' \vdash new\vec{\alpha}_n.t \sim\mathcal{R}\sim new\vec{\alpha}_n.u : \delta^n\mathbb{N} \to \mathbb{P}$. By multiple applications of Lemma 3.5 we derive $s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.\lambda\zeta.[\![P']\!] : \delta^n(\mathbb{N} \to \mathbb{P})$ from $s' \mathbin{\dot\cup} \vec{a} \vdash t[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. We also derive $s' \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.\lambda\zeta.[\![Q']\!] : \delta^n(\mathbb{N} \to \mathbb{P})$ from $s' \mathbin{\dot\cup} \vec{a} \vdash u[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. Since $P' \sim_l Q'$, by construction of $\mathcal{R}$, we finally have

$$s' \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.\lambda\zeta.[\![P']\!] \;\mathcal{R}\; new\vec{\alpha}.\lambda\zeta[\![Q']\!] \sim new\vec{\alpha}.u : \delta^n(\mathbb{N} \to \mathbb{P}) \;.$$

This concludes the analysis. $\square$

Theorem A.19 *Let P and Q two closed π-calculus processes. If $\mathsf{n}(P,Q) \vdash [\![P]\!] \sim [\![Q]\!] : \mathbb{P}$, then $P \sim_l Q$.*

Proof We prove a stronger theorem:

> Let P and Q two π-calculus processes such that $\mathrm{fv}(P) = \mathrm{fv}(Q) = \vec{\alpha}_n$. If $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}_n.[\![P]\!] \sim new\vec{\alpha}_n.[\![Q]\!] : \delta^n\mathbb{P}$, then $P \sim_l Q$.

Let

$$\mathcal{R} \;=\; \{(P,Q) \mid \mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}_n.[\![P]\!] \sim new\vec{\alpha}_n.[\![Q]\!] : \delta^n\mathbb{P} \text{ and } \mathrm{fv}(P) = \mathrm{fv}(Q) = \alpha_n\} \;.$$

We prove that $\mathcal{R}$ is a strong late bisimulation. Suppose $P \;\mathcal{R}\; Q$ and $P \xrightarrow{\ell} P'$. We perform a case analysis on ℓ.

- Suppose that $P \xrightarrow{\tau} P'$. Since transition are preserved by bijective substitution, we have $P[\vec{a}/\vec{\alpha}] \xrightarrow{\tau} P'[\vec{a}/\vec{\alpha}]$, for a set $\vec{a}$ of names fresh for both for P and Q. By Lemma A.17, $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\tau:!} t[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash t[\vec{a}/\vec{\alpha}] \sim [\![P']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. By the operational rules we get the transition $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.[\![P]\!] \xrightarrow{new\vec{\alpha}.\tau:!} new\vec{\alpha}.t$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.[\![P']\!] : \delta^n\mathbb{P}$. By bisimulation, there is a transition $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.[\![Q]\!] \xrightarrow{new\vec{\alpha}.\tau:!} new\vec{\alpha}.u$, with $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.u : \delta\mathbb{P}$. This must have been derived from $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\tau:!} u[\vec{a}/\vec{\alpha}]$, and by Corollary 3.7 we have $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash t[\vec{a}/\vec{\alpha}] \sim u[\vec{a}/\vec{\alpha}] : \mathbb{P}$. By Lemma A.17, $Q[\vec{a}/\vec{\alpha}] \xrightarrow{\tau} Q'[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash u[\vec{a}/\vec{\alpha}] \sim [\![Q']\!][\vec{a}/\vec{\alpha}] : \mathbb{P}$. In turn $Q \xrightarrow{\tau} Q'$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.[\![Q']\!] : \delta^n\mathbb{P}$. As $\sim$ is transitive, by construction of $\mathcal{R}$ we conclude $P' \;\mathcal{R}\; Q'$.

- The case $P \xrightarrow{\overline{n}m}$ is similar to the previous one.

- Suppose $P \xrightarrow{\overline{n}(\zeta)} P'$. Since transition are preserved by bijective substitution, we have $P[\vec{a}/\vec{\alpha}] \xrightarrow{\overline{e}(\zeta)} P'[\vec{a}/\vec{\alpha}]$, for $\alpha = \mathrm{fn}(P)$, $\vec{a}$ a set of names fresh for both for P and Q, and $e = a_i$ if $n = \alpha_i$ and $e = n$ otherwise. Observe that $\zeta \notin \vec{\alpha}$. By Lemma A.17, $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{bout}:e\cdot!} t[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs,rhs) \mathbin{\dot\cup} \{\vec{a}\} \vdash t[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. By the operational rules we get $\mathsf{n}(lhs,rhs) \vdash new\vec{\alpha}.[\![P]\!] \xrightarrow{new\vec{\alpha}.\mathsf{bout}:n\cdot!} new\vec{\alpha}.t$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.new\zeta.[\![P']\!] : \delta^n\delta\mathbb{P}$. By bisimulation, $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.[\![Q]\!] \xrightarrow{new\vec{\alpha}.\mathsf{bout}:n\cdot !} new\vec{\alpha}.u$, with $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.u : \delta^n\delta\mathbb{P}$. This must have been derived from $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{bout}:e\cdot !} u[\vec{a}/\vec{\alpha}]$, and by Corollary 3.7 we have $\mathsf{n}(lhs, rhs) \vdash t[\vec{a}/\vec{\alpha}] \sim u[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. By Lemma A.17, $Q[\vec{a}/\vec{\alpha}] \xrightarrow{\overline{e}(\zeta)} Q'[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash u[\vec{a}/\vec{\alpha}] \sim (new\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \delta\mathbb{P}$. In turn $Q \xrightarrow{\overline{n}(\zeta)} Q'$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.u \sim new\vec{\alpha}.new\zeta.[\![Q']\!] : \delta^n\delta\mathbb{P}$. As $\sim$ is transitive, by construction of $\mathcal{R}$ we conclude $P' \mathcal{R} Q'$.

- Suppose $P \xrightarrow{n\zeta} P'$. Since transition are preserved by bijective substitution, we have $P[\vec{a}/\vec{\alpha}] \xrightarrow{e\zeta} P'[\vec{a}/\vec{\alpha}]$, for $\alpha = \mathrm{fn}(P)$, $\vec{a}$ a set of names fresh for both for P and Q, and $e = a_i$ if $n = \alpha_i$ and $e = n$ otherwise. Observe that $\zeta \notin \vec{\alpha}$. By Lemma A.17, $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash [\![P]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{inp}:e\cdot !} t[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash t[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. By the operational rules we get $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.[\![P]\!] \xrightarrow{new\vec{\alpha}.\mathsf{inp}:n\cdot !} new\vec{\alpha}.t$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.\lambda\zeta.[\![P']\!] : \delta^n\mathbb{N} \to \mathbb{P}$. By bisimulation, $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.[\![Q]\!] \xrightarrow{new\vec{\alpha}.\mathsf{inp}:n\cdot !} new\vec{\alpha}.u$, with $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.u : \delta^n\mathbb{N} \to \mathbb{P}$. This must have been derived from $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash [\![Q]\!][\vec{a}/\vec{\alpha}] \xrightarrow{\mathsf{inp}:e\cdot !} u[\vec{a}/\vec{\alpha}]$, and by Corollary 3.7 we have $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash new\vec{\alpha}.t \sim new\vec{\alpha}.u : \delta^n\mathbb{N} \to \mathbb{P}$. By Lemma A.17, $Q[\vec{a}/\vec{\alpha}] \xrightarrow{e\zeta} Q'[\vec{a}/\vec{\alpha}]$, with $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash u[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. In turn $Q \xrightarrow{n\zeta} Q'$, and by multiple applications of Lemma 3.5 we have $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.\lambda\zeta.P' \sim new\vec{\alpha}.\lambda\zeta.[\![Q']\!] : \delta^n(\mathbb{N} \to \mathbb{P})$. It remains to prove that for all m it holds $P'[m/\zeta] \mathcal{R} Q'[m/\zeta]$. By transitivity of $\sim$ we have $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.\lambda\zeta.[\![P']\!] \sim new\vec{\alpha}.\lambda\zeta.[\![Q']\!] : \delta^n\mathbb{N} \to \mathbb{P}$ and by multiple applications of Corollary 3.7 we derive $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash (\lambda\zeta.[\![P']\!])[\vec{a}/\vec{\alpha}] \sim (\lambda\zeta.[\![Q']\!])[\vec{a}/\vec{\alpha}] : \mathbb{N} \to \mathbb{P}$. We perform a case analysis on m.
  - If m is a name constant, then by Lemma A.1, (repeated applications of) Corollary 3.8, and Proposition 3.3, there is a $\vec{a}'$ such that $m \cap a' = \emptyset$ and $(\mathsf{n}(lhs, rhs) \cup \{m\}) \dot\cup \{\vec{a}'\} \vdash [\![P']\!][\vec{a}'/\vec{\alpha}][m/\zeta] \sim [\![Q']\!][\vec{a}'/\vec{\alpha}][m/\zeta] : \mathbb{P}$. By multiple applications of Lemma 3.5 we obtain $\mathsf{n}(lhs, rhs) \cup \{m\} \vdash new\vec{\alpha}.[\![P'[m/\zeta]]\!] \sim new\vec{\alpha}.[\![Q'[m/\zeta]]\!] : \delta^n\mathbb{P}$ and by construction of $\mathcal{R}$ we get $P'[m/\zeta] \sim_l Q'[m/\zeta]$.
  - Suppose that m is a name variable and $m \notin \vec{\alpha}$. By congruence, the equation $(\mathsf{n}(lhs, rhs)) \dot\cup \{\vec{a}'\} \vdash newm.((\lambda\zeta.[\![P']\!])m)[\vec{a}'/\vec{\alpha}] \sim newm.((\lambda\zeta.[\![Q']\!])m)[\vec{a}'/\vec{\alpha}] : \delta\mathbb{P}$ holds. By Proposition 3.3 and by by multiple applications of Lemma 3.5 we obtain $\mathsf{n}(lhs, rhs) \vdash new\vec{\alpha}.newm.[\![P'[m/\zeta]]\!] \sim new\vec{\alpha}.newm.[\![Q'[m/\zeta]]\!] : \delta^n\delta\mathbb{P}$ and by construction of $\mathcal{R}$ we get $P'[m/\zeta] \sim_l Q'[m/\zeta]$.
  - Finally, suppose that m is a name variable and $m = \alpha_i$. By Proposition 3.3 we have $\mathsf{n}(lhs, rhs) \dot\cup \{\vec{a}\} \vdash [\![P']\!][\vec{a}'/\vec{\alpha}][a_i/\zeta] \sim [\![Q']\!][\vec{a}'/\vec{\alpha}][a_i/\zeta] : \mathbb{P}$. By multiple applications of Lemma 3.5 we obtain $\mathsf{n}(lhs, rhs) \cup \{m\} \vdash new\vec{\alpha}.[\![P'[\alpha_i/\zeta]]\!] \sim new\vec{\alpha}.[\![Q'[\alpha_i/\zeta]]\!] : \delta^n\mathbb{P}$ and by construction of $\mathcal{R}$ we get $P'[m/\zeta] \sim_l Q'[m/\zeta]$.

This concludes the analysis. $\square$

Recent BRICS Report Series Publications

RS-04-21 Glynn Winskel and Francesco Zappa Nardelli. *New-HOPLA—A Higher-Order Process Language with Name Generation*. October 2004. 38 pp.

RS-04-20 Mads Sig Ager. *From Natural Semantics to Abstract Machines*. October 2004. 21 pp. Presented at the *International Symposium on Logic-based Program Synthesis and Transformation*, LOPSTR 2004, Verona, Italy, August 26–28, 2004.

RS-04-19 Bolette Ammitzbøll Madsen and Peter Rossmanith. *Maximum Exact Satisfiability: NP-completeness Proofs and Exact Algorithms*. October 2004. 20 pp.

RS-04-18 Bolette Ammitzbøll Madsen. *An Algorithm for Exact Satisfiability Analysed with the Number of Clauses as Parameter*. September 2004. 4 pp.

RS-04-17 Mayer Goldberg. *Computing Logarithms Digit-by-Digit*. September 2004. 6 pp.

RS-04-16 Karl Krukow and Andrew Twigg. *Distributed Approximation of Fixed-Points in Trust Structures*. September 2004. 25 pp.

RS-04-15 Jesús Fernando Almansa. *Full Abstraction of the UC Framework in the Probabilistic Polynomial-time Calculus ppc*. August 2004.

RS-04-14 Jesper Makholm Byskov. *Maker-Maker and Maker-Breaker Games are PSPACE-Complete*. August 2004. 5 pp.

RS-04-13 Jens Groth and Gorm Salomonsen. *Strong Privacy Protection in Electronic Voting*. July 2004. 12 pp. Preliminary abstract presented at Tjoa and Wagner, editors, *13th International Workshop on Database and Expert Systems Applications*, DEXA '02 Proceedings, 2002, page 436.

RS-04-12 Olivier Danvy and Ulrik P. Schultz. *Lambda-Lifting in Quadratic Time*. June 2004. 34 pp. To appear in *Journal of Functional and Logic Programming*. This report supersedes the earlier BRICS report RS-03-36 which was an extended version of a paper appearing in Hu and Rodríguez-Artalejo, editors, *Sixth International Symposium on Functional and Logic Programming*, FLOPS '02 Proceedings, LNCS 2441, 2002, pages 134–151.