

Nominal π -calculus and nominal automata

Nikos Tzevelekos

Oxford University Computing Laboratory
nikt@comlab.ox.ac.uk

Abstract. In this paper we introduce a nominal π -calculus and examine its operational behavior under early-transition relation. Our calculus is a reformulation of the $\pi\theta$ -calculus, i.e. the π -calculus of name-abstracted processes, in the theory of nominal sets of Gabbay and Pitts. The mechanics of nominal sets provide ease of nominal manipulations such as name-abstractions, and the use of abstracted processes yields a finitely-branching transition relation. Moreover, we introduce nominal automata as the natural candidates for representing the operational semantics.

1 Introduction

The treatment of names in computational theories has received notable attention since researchers ceased to be satisfied with the conventional approach, epitomised in the *Barendregt variable convention*. It was argued (see e.g. [MP99,Pit03]) that the conventional approach of name binding in abstract syntax, which equates terms under α -equivalence and then reasons with specifically selected *representatives*, leaves a lot to be desired: an abstract syntax involving α -equivalence classes as base terms is more appropriate. Such a syntax is provided by use of (at least) two kinds of abstract syntax: Higher-Order Abstract Syntax and Nominal Abstract Syntax. In this paper we examine the use of the latter for a presentation of the π -calculus¹.

We formulate a nominal π -calculus based on the theory of nominal sets developed by Gabbay and Pitts [Gab00,GP02,Pit03]. Our presentation deals elegantly with α -equivalence, name-freshness and name-abstraction, and introduces a transition relation which efficiently encodes early transition. An earlier investigation on the π -calculus in this nominal setting was carried out in [Gab03], where early and late transition relations were considered. In this paper we reformulate that calculus by considering abstracted² processes and actions (and adding recursion), which allows us to use simpler notions of nominal theory³.

The use of abstracted processes in the π -calculus is not new. Apart from work conducted in the HOAS framework (e.g. [MP99,HMS01]), in [BHLM04] the $\pi\theta$ -calculus is introduced as a calculus of “processes under a list of abstractions”, these latter called θ -abstractions. The operational semantics of $\pi\theta$ is

¹ And refer the reader to e.g. [GP02, Section 1.1] and [Gab03, Section 5] for a discussion on the pros and cons of typing nominal signatures in NAS instead of HOAS.

² We refer to nominal abstraction, see definition 2.

³ Specifically, we do not use generalised \mathcal{N} -quantifiers and related notions.

finitely branching and allows for a finite representation of finitary processes in appropriate automata: θ -automata. Our $\mathcal{N}\pi$ -calculus is a nominalisation of $\pi\theta$.

As in [BHLM04], the transition relation we use is ‘economically’ early, by behaving early when receiving *known names* and late when receiving (or creating) *fresh ones* (definition 13). In contrast to [BHLM04], the notion of known names is taken directly from the processes (i.e. from their *finite support*), instead of explicitly attaching it to the transition relation. The resulting transition relation is finitely branching and allows for efficient algorithmic interpretations of early transition in *nominal automata*, which we introduce.

Nominal automata (or \mathcal{N} -automata) are automata built in nominal sets, and which use a memory list (called a *proper name-list*) to store names. At each step of the computation the automaton evaluates its input by recourse to its current name-list, and the list is updated at each such step. List update involves introduction of fresh names (which is modeled as name-abstraction) or deletion of known names that are not of further use. \mathcal{N} -automata are more general than θ -automata, as the latter are too specific to π -calculus processes (e.g. in order to check indexed bisimilarity of two θ -automata one has to check transitions of $\pi\theta$ -processes not appearing in the automata), and could be seen as an analog of HD-automata ([MP05]) in nominal sets, yet with the additional feature of using also abstracted states (as θ -automata do).

2 Nominal sets, the category \mathbf{Nom}

In this paper all constructions are based on Nominal Sets ([GP02,Pit03]), which are ordinary sets entailing names (*atoms*), in the following manner.

Definition 1 (Nom, # and \mathcal{N}) We assume a distinguished countably infinite set of names \mathbf{N} . A nominal set X is a set equipped with a name-swapping operation $(- \ -) \cdot - : \mathbf{N}^2 \times X \rightarrow X$ such that for all $\alpha, \alpha', \beta, \beta' \in \mathbf{N}$, $x \in X$,

$$(\alpha \ \alpha) \cdot x = x, \ (\alpha \ \beta) \cdot (\alpha \ \beta) \cdot x = x \text{ and } (\alpha \ \beta) \cdot (\alpha' \ \beta') \cdot x = ((\alpha \ \beta) \cdot \alpha' \ (\alpha \ \beta) \cdot \beta') \cdot (\alpha \ \beta) \cdot x$$

Moreover, all elements of $x \in X$ have finite support $\mathbf{S}(x)$, where

$$\mathbf{S}(x) := \{\alpha \mid \text{for infinitely many } \beta. (\alpha \ \beta) \cdot x \neq x\}$$

x is *equivariant* if it has empty support. We say that α is *fresh for* x , and write $\alpha \# x$, if $\alpha \notin \mathbf{S}(x)$. A function $f : X \rightarrow Y$ between nominal sets X, Y is called *equivariant* if, for every $\alpha, \beta \in \mathbf{N}$ and $x \in X$, $(\alpha \ \beta) \cdot f(x) = f((\alpha \ \beta) \cdot x)$. Thus,

Nom is the category of nominal sets and equivariant functions. \triangle

So \mathbf{N} in particular is a nominal set, and so is $\mathbf{N}^\#$, the set of finite lists of distinct names. Moreover, $(- \ -) \cdot - : \mathbf{N}^2 \times X \rightarrow X$ are by definition equivariant. We also introduce the *freshness quantifier*:

$$\forall \alpha \in \mathbf{N}. \phi \iff \phi \text{ holds for cofinitely many } \alpha \in \mathbf{N}$$

Some further definitions will be of use.

Definition 2 Let X be a nominal set, $x \in X$ and $\alpha \in \mathbf{N}$, and define:

- $\mathbf{N}\#X := \{(\alpha, x) \in \mathbf{N} \times X \mid \alpha\#x\}$ and $\mathbf{N}\#X := \{(\alpha, x) \in \mathbf{N} \times X \mid \alpha\#x\}$.
- The **abstraction** of x at α as

$$\langle \alpha \rangle x := \{(\beta, (\alpha \beta) \cdot x) \mid \beta = \alpha \vee \beta\#x\}$$

We have $\mathbf{S}(\langle \alpha \rangle x) = \mathbf{S}(x) \setminus \{\alpha\}$, and setting $\langle \mathbf{N} \rangle X := \{\langle \alpha \rangle x \mid (\alpha, x) \in \mathbf{N} \times X\}$ we get $\langle _ \rangle _ : \mathbf{N} \times X \rightarrow \langle \mathbf{N} \rangle X$ in **Nom**.

Given some basic nominal set X_0 we obtain the abstracted sets:

$$\langle \mathbf{N} \rangle^0 X_0 := X_0, \quad \langle \mathbf{N} \rangle^{n+1} X_0 := \langle \mathbf{N} \rangle \langle \mathbf{N} \rangle^n X_0, \quad \langle \mathbf{N} \rangle^* X_0 := \bigcup_{n \in \omega} \langle \mathbf{N} \rangle^n X_0$$

Usually, when we use metavariable x to range in $\langle \mathbf{N} \rangle^* X_0$ then \tilde{x} ranges in X_0 and \hat{x} ranges in $\langle \mathbf{N} \rangle^{>0} X_0$.

- Dually to name-abstraction, **name-concretion** $@ : \mathbf{N}\#\langle \mathbf{N} \rangle X \rightarrow X$. For each $(\alpha, \hat{x}) \in \mathbf{N}\#\langle \mathbf{N} \rangle X$ the concretion of \hat{x} at α is the unique $y \in X$ such that $\hat{x} = \langle \alpha \rangle y$, i.e. $\hat{x}@ \alpha = y \iff \hat{x} = \langle \alpha \rangle y$.
- The nominal set of **strict name-abstractions** on X , $\langle \mathbf{N} \rangle X := \{\langle \alpha \rangle x \mid \alpha\#x\}$, and, as above, $\langle \mathbf{N} \rangle^n X$ and $\langle \mathbf{N} \rangle^* X$. \triangle

Regarding notation, note that we will usually denote sequences of names $\alpha_1 \dots \alpha_n$ by $\vec{\alpha}$, and we will write $\langle \vec{\alpha} \rangle x$ for $\langle \alpha_1 \rangle \dots \langle \alpha_n \rangle x$, and $x@ \vec{\alpha}$ for $(\dots(x@ \alpha_1)@ \dots)@ \alpha_n$.

A useful nominal notion, introduced in [Gab00] as fresh operator, is that of **cofinitely constant values**: if $f : \mathbf{N}\#X \rightarrow Y$ returns, at any $x \in X$, the same $f(\alpha, x)$ for cofinitely many $\alpha \in \mathbf{N}$, then we can obtain a ‘curried’ version of f , denoted $\forall \alpha \cdot f(\alpha, _) : X \rightarrow Y$, returning for each x precisely this (unique) value.

Proposition 3 Let $f : \mathbf{N}\#X \rightarrow Y$ in **Nom**. There exists a unique morphism $\tilde{f} : X \rightarrow Y$ s.t. $\forall x \in X. \forall \alpha \in \mathbf{N}. \tilde{f}(x) = f(\alpha, x)$, iff $\forall x \in X. \forall \alpha \in \mathbf{N}. \alpha\#f(\alpha, x)$. \square

Definition 4 (CCV) Let $f : \mathbf{N}\#X \rightarrow Y$ in **Nom**. If f satisfies the condition:

$$\forall x \in X. \forall \alpha \in \mathbf{N}. \alpha\#f(\alpha, x) \quad (\text{CCV})$$

then define, for each $x \in X$, $\forall \alpha \cdot f(\alpha, x)$ to be the value $\tilde{f}(x)$, with \tilde{f} as in the above proposition. $\forall \alpha \cdot f(\alpha, x)$ is the **cofinitely constant value** of the non-equivariant partial function $f(_, x)$. \triangle

This operator gives us all the machinery needed to reason about abstracted terms **namelessly**, i.e. without having to resort to selection of representatives. For example, we can extend name-abstraction into a functor $\langle \mathbf{N} \rangle : \mathbf{Nom} \rightarrow \mathbf{Nom}$ by sending each $f : X \rightarrow Y$ to $\langle \mathbf{N} \rangle f : \langle \mathbf{N} \rangle X \rightarrow \langle \mathbf{N} \rangle Y$, where $\langle \mathbf{N} \rangle f(\hat{x}) := \forall \alpha \cdot \langle \alpha \rangle f(\hat{x}@ \alpha)$. Functoriality of $\langle \mathbf{N} \rangle$ can be easily shown using the following proposition.

Proposition 5 Let $f : \mathbf{N}\#(Y \times Z) \rightarrow W$ and $g : \mathbf{N}\#X \rightarrow Z$ in **Nom**. If $\forall x \in X. \forall \alpha \in \mathbf{N}. \alpha\#g(\alpha, x)$ and $\forall y \in Y. \forall z \in Z. \forall \beta \in \mathbf{N}. \beta\#f(\beta, y, z)$ then we have, for all $x \in X, y \in Y, \forall \beta \cdot f(\beta, y, \forall \alpha \cdot g(\alpha, x)) = \forall \beta \cdot f(\beta, y, g(\beta, x))$ \square

2.1 Proper name-lists

⁴ A construction in **Nom** particularly useful to us is that of proper name-lists, which are a special class of abstracted lists of names. Given a basic nominal set X_0 , we can describe any element $x \in \langle \mathbf{N} \rangle^* X_0$ by a pair (L, x') of a proper name-list and an element of $\langle \mathbf{N} \rangle^* X_0$, so that L contains precisely the abstraction-structure of x whereas x' contains only the strict abstractions. This description will be useful when considering algorithmic interpretations of the nominal π -calculus, so in place of X_0 we put some *set of basic states* St_{s_0} .

Definition 6 (Proper name-lists) Define $\mathbb{L}_0 := \mathbf{N}^\#$, and fix the denotation $\check{L} = [\check{L}_1, \dots, \check{L}_n]$, for each $\check{L} \in \mathbb{L}_0$. Proper name-lists are abstracted name-lists $L \in \langle \mathbf{N} \rangle^n \mathbb{L}_0$ which satisfy the *properness condition*: $\langle \vec{\alpha} \rangle \check{L}$ is proper iff

$$\forall i, j, i', j'. \alpha_i = \check{L}_{i'} \wedge \alpha_j = \check{L}_{j'} \wedge i < j \implies i' < j'$$

We let $\mathbb{L} \subseteq \langle \mathbf{N} \rangle^* \mathbb{L}_0$ denote the set of proper name-lists. Δ

Informally, proper name-lists are lists of distinct names with name-abstractions on the front that respect the list-ordering. For example, let:

$$\check{P} \in St_{s_0}, \mathbf{S}(\check{P}) = \{x, z\}, P := \langle xyz \rangle \check{P}, L := \langle xyz \rangle [x, z] \text{ and } P' := \langle xz \rangle \check{P} \quad (1)$$

Then, P can be represented by $(P', L) \in (\langle \mathbf{N} \rangle^* St_{s_0}) \times \mathbb{L}$.

We proceed to some useful functions for proper-name-list manipulation.

Definition 7 Let $\check{L} = [\alpha_1, \dots, \alpha_n] \in \mathbb{L}_0$ and $L \in \mathbb{L}$, and define:

- A length function $|\cdot| : \mathbb{L} \rightarrow \omega$ by: $|\check{L}|_0 := n$, $|L| := \mathcal{V} \vec{\alpha} \cdot |L @ \vec{\alpha}|_0$.
- Left-push and right-push functions **push** and **add**, both of type $\mathbf{N} \# \mathbb{L} \rightarrow \mathbb{L}$:

$$\begin{aligned} \text{push}_0(x, \check{L}) &:= [x, \alpha_1, \dots, \alpha_n] & \text{push}(x, L) &:= \mathcal{V} \vec{\alpha} \cdot \langle \vec{\alpha} \rangle \text{push}_0(x, L @ \vec{\alpha}) \\ \text{add}_0(x, \check{L}) &:= [\alpha_1, \dots, \alpha_n, x] & \text{add}(x, L) &:= \mathcal{V} \vec{\alpha} \cdot \langle \vec{\alpha} \rangle \text{add}_0(x, L @ \vec{\alpha}) \end{aligned}$$

- Fresh-name addition function **addfresh** : $\mathbb{L} \rightarrow \mathbb{L}$, which right-pushes some fresh name in a list and inner-abstracts on it:

$$\text{addfresh}(L) := \mathcal{V} \vec{\alpha} \cdot \langle \vec{\alpha} \rangle \mathcal{V} \beta \cdot \langle \beta \rangle \text{add}_0(\beta, L @ \vec{\alpha})$$

- A deletion function **del** : $\{\vec{m} \in \omega^* \mid \vec{m} \text{ strictly increasing}\} \rightarrow \mathbb{L} \rightarrow \mathbb{L}$:

$$\text{del}(\vec{m})_0(\check{L}) := \check{L} - [\alpha_{m_1}, \dots, \alpha_{m_{|\vec{m}|}}] \quad , \quad \text{del}(\vec{m})(L) := \mathcal{V} \vec{\alpha} \cdot \langle \vec{\alpha} \rangle \text{del}(\vec{m})_0(L @ \vec{\alpha})$$

(where we write $L - L'$ for sublist deletion). We let $|\text{del}(\vec{m})| := |\vec{m}|$. Δ

Not all pairs of lists and strictly-abstracted states can represent a state, and not all pairs of lists and states are such that the list encodes the abstraction-structure of the state; some compatibility constraints must be satisfied.

⁴ Note that the material in this section won't be needed until section 3.1.

Definition 8 Let Sts_0 be some basic nominal set of states, $Sts^s := \langle \mathbb{N} \rangle^* Sts_0$ and $Sts := \langle \mathbb{N} \rangle^* Sts_0$. The compatible Sts^s - \mathbb{L} pairs form the set $Sts^s \boxtimes \mathbb{L}$ and the compatible Sts - \mathbb{L} pairs form $Sts \boxtimes \mathbb{L}$, as follows.

$$\begin{aligned} Sts^s \boxtimes \mathbb{L} &:= \{(P, L) \in Sts^s \times \mathbb{L} \mid \mathfrak{S}(P) \cap \mathfrak{S}(L) = \emptyset \wedge \exists m. P \in \langle \mathbb{N} \rangle^m Sts_0 \wedge m = |L|\} \\ Sts \boxtimes \mathbb{L} &:= \{(P, L) \in Sts \times \mathbb{L} \mid \mathfrak{S}(L) \subseteq \mathfrak{S}(P) \wedge \forall \vec{\alpha} \in \mathbb{N}. \forall i \leq |\vec{\alpha}|. \alpha_i \# P @ \vec{\alpha} \Leftrightarrow \alpha_i \# L @ \vec{\alpha}\} \end{aligned}$$

For each $P \in Sts$ we obtain an equivariant proper name-list $(P)^\square$ and a strictly-abstracted state $(P)^\boxtimes$ as follows. If $P \in Sts_0$ then $(P)^\square := []$ and $(P)^\boxtimes := P$, ow:

$$(P)^\square := \begin{cases} \forall \alpha. \langle \alpha \rangle (P @ \alpha)^\square & \\ \forall \alpha. \langle \alpha \rangle \text{push}(\alpha, (P @ \alpha)^\square) & \end{cases} \quad (P)^\boxtimes := \begin{cases} \forall \alpha. (P @ \alpha) & \text{if } \forall \alpha \in \mathbb{N}. \alpha \# P @ \alpha \\ \forall \alpha. \langle \alpha \rangle (P @ \alpha) & \text{if } \forall \alpha \in \mathbb{N}. \alpha \# P @ \alpha \end{cases}$$

△

The compatibility constraints of $Sts \boxtimes \mathbb{L}$ ensure that, in a pair $(P, L) \in Sts \boxtimes \mathbb{L}$, P can be described by a strictly-abstracted state $P' \in Sts^s$ and the list L , as they ensure that the list L follows the abstraction-structure of state P . Notably, $(P, (P)^\square) \in Sts \boxtimes \mathbb{L}$, since $(P)^\square$ is constructed precisely by copying the abstraction-structure of P . On the other hand, the compatibility constraints of $Sts^s \boxtimes \mathbb{L}$ ensure that a pair $(P, L) \in Sts^s \boxtimes \mathbb{L}$ may encode a state $P' \in Sts$: P and L must not have common names, while L may keep enlisted only those abstracted names that are active in P . In particular, $((P)^\boxtimes, (P)^\square) \in Sts^s \boxtimes \mathbb{L}$, where $(P)^\square$ is just P with its non-strict abstractions removed.

We formalise the intuitions regarding representations of abstracted states by pairs of strictly-abstracted states and lists with the following translations.

Definition 9 Let Sts_0 be a basic nominal set of states, and Sts, Sts^s as above. Define the translations $(-)^\boxtimes : Sts \boxtimes \mathbb{L} \rightarrow Sts^s$ and $(-)^\square : Sts^s \boxtimes \mathbb{L} \rightarrow Sts$ as:

$$(P, L)^\boxtimes := \begin{cases} \langle L_1 \dots L_n \rangle P & \\ \forall \alpha. (P @ \alpha, L @ \alpha)^\boxtimes & \end{cases} \quad (P, L)^\square := \begin{cases} P @ (L_1 \dots L_n) & , \text{ if } L \in \mathbb{L}_0 \\ \forall \alpha. \langle \alpha \rangle (P, L @ \alpha)^\square & , \text{ otherwise} \end{cases}$$

△

Note that, for any $(P, L) \in Sts \boxtimes \mathbb{L}$, we have $((P, L)^\boxtimes, L) \in Sts^s \boxtimes \mathbb{L}$, and, for any $(P, L) \in Sts^s \boxtimes \mathbb{L}$, we have $((P, L)^\square, L) \in Sts \boxtimes \mathbb{L}$. Moreover, for any $P \in Sts$, $(P)^\boxtimes = (P, (P)^\square)^\boxtimes$. The relation between the above translations is depicted by the following proposition.

Proposition 10 Let Sts_0 be some basic set of states, and Sts, Sts^s as above. Then, for each $(P, L) \in Sts \boxtimes \mathbb{L}$, $((P, L)^\boxtimes, L)^\square = P$, and, for each $(Q, L) \in Sts^s \boxtimes \mathbb{L}$, $((Q, L)^\square, L)^\boxtimes = Q$. □

Hence, returning to our previous example (1), we see that $(P, L) \in Sts \boxtimes \mathbb{L}$ and $(P', L) \in Sts^s \boxtimes \mathbb{L}$. Moreover, $L = (P)^\square$, $(P, L)^\boxtimes = P'$ and $(P', L)^\square = P$.

3 The Nominal π -calculus ($\mathcal{N}\pi$ -calculus)

Let us fix a countably infinite set \mathbf{N} of names $x, y, \alpha, \beta, \dots$, and consider nominal sets over \mathbf{N} . The nominal π -calculus ($\mathcal{N}\pi$ -calculus) is then defined as follows.

Definition 11 The processes of the $\mathcal{N}\pi$ -calculus form the nominal set Π :

$$\Pi := \langle \mathbf{N} \rangle^* \Pi_0 \quad \text{where}$$

$$\Pi_0 \ni \check{P} ::= \mathbf{0} \mid \tau.\check{P} \mid \bar{x}y.\check{P} \mid x.\langle y \rangle \check{P} \mid \check{P} \mid \check{P} \mid \check{P} + \check{P} \mid \nu \langle y \rangle \check{P} \mid [x = y] \check{P} \mid p(\bar{x})$$

Elements of Π are ranged over by $P, Q, \text{etc.}$ Each used process constant p must be accompanied by a *definition* of the form $p(\bar{y}) \triangleq P$, where $P \in \Pi_0$ and $\mathbf{S}(P) = \mathbf{S}(\bar{y})$ and $\bar{y} \in \mathbf{N}^\#$. Moreover, each occurrence of a process variable p must be *guarded*, i.e. must come in one of the forms $\bar{x}y.p$, $x.\langle y \rangle p$ or $\tau.p$. \triangle

In comparison to the ordinary π -calculus one notes the formulations for input processes ($x.\langle y \rangle P$) and fresh-name processes ($\nu \langle x \rangle P$), as well as the use of abstracted processes ($\hat{P} \in \langle \mathbf{N} \rangle^{n>0} \Pi_0$), which stand for (α -equivalence classes of) processes having received or created fresh names.

Although the decorated notation \hat{P} is useful for the definition, we will abandon it for most of the sequel for aesthetic reasons. Now, we extend constructions for parallel composition and ν -abstractions to processes in $\langle \mathbf{N} \rangle \Pi_0$ as follows. For any $\hat{P}, \hat{Q} \in \langle \mathbf{N} \rangle \Pi_0$ and $P, Q \in \Pi_0$,

$$\begin{aligned} \hat{P} \mid \hat{Q} &:= \mathcal{N}\alpha . \langle \alpha \rangle (\hat{P} @ \alpha \mid \hat{Q} @ \alpha) & \hat{P} \mid Q &:= \mathcal{N}\alpha . \langle \alpha \rangle (\hat{P} @ \alpha \mid Q) \\ P \mid \hat{Q} &:= \mathcal{N}\alpha . \langle \alpha \rangle (P \mid \hat{Q} @ \alpha) & \nu \langle x \rangle \hat{P} &:= \mathcal{N}\alpha . \langle \alpha \rangle \nu \langle x \rangle (\hat{P} @ \alpha) \end{aligned} \quad (2)$$

These will be used implicitly below (e.g. in PAR, CLOSE, RES of definition 13).

Name-swapping comes for free in Π , as it is a nominal set. It follows that the support of a process P is exactly the set of its “free names”. Name-for-name substitution (simply *substitution*) in Π can be introduced as an arrow in **Nom**, $\{-/-\}_- : \mathbf{N}^2 \times \Pi \rightarrow \Pi$. For any $P \in \Pi$ and any $x_1, x_2 \in \mathbf{N}$, let σ be $\{x_1/x_2\}$ and define $\sigma \cdot P := \mathcal{N}\bar{\alpha} . \langle \bar{\alpha} \rangle \sigma \cdot (P @ \bar{\alpha})$, where the case of $P \in \Pi_0$ is given by:

$$\sigma \cdot \mathbf{0} := \mathbf{0} \quad \sigma \cdot (\tau.P) := \tau . \sigma \cdot P \quad \sigma \cdot (x.\hat{P}) := (\sigma \cdot x) . \sigma \cdot \hat{P} \quad [\dots]$$

and $\{x_1/x_2\} \cdot \hat{P} := \mathcal{N}\alpha . \langle \alpha \rangle \{x_1/x_2\} \cdot (\hat{P} @ \alpha)$, the unique process we come out with for every $\alpha \# x_1, x_2, \hat{P}$, if we do all the syntactic operations in $\langle \alpha \rangle \{x_1/x_2\} \cdot (\hat{P} @ \alpha)$.

Using substitution we can expand name-concretion to non-fresh names. That is, we define (equivariant) $@ : \mathbf{N} \times \langle \mathbf{N} \rangle \Pi \rightarrow \Pi$ as:

$$\hat{P} @ \beta := \mathcal{N}\alpha . \{\beta/\alpha\} \cdot (\hat{P} @ \alpha) \quad (3)$$

Substitutions distribute over $@$. Moreover, $\#$ is reflected by substitution (but not preserved by it, e.g. $\{\alpha/\beta\} \cdot \beta \# \{\alpha/\beta\} \cdot \alpha$).

Proposition 12 For any substitution σ , any $y \in \mathbf{N}$, $P \in \Pi$ and $\hat{P} \in \Pi \setminus \Pi_0$,

- $\mathbf{S}(\sigma \cdot P) = \sigma \cdot \mathbf{S}(P)$ and $y \# P \implies \sigma \cdot y \# \sigma \cdot P$
- $(\sigma \cdot \hat{P}) @ (\sigma \cdot y) = \sigma \cdot (\hat{P} @ y)$ □

We proceed to the transition calculus of $\mathcal{V}\pi$.

Definition 13 The set of actions is a nominal set Act (ranged over by a, b , etc.):

$$Act := \langle \mathbb{N} \rangle^* Act_0 \quad \text{where} \quad Act_0 \ni \check{a} ::= \tau \mid \bar{x}y \mid xy \mid x\bullet \mid \bar{x}\circ$$

τ, \bullet and \circ are equivariant constants. $x\bullet$ denotes input of a locally fresh name, and $\bar{x}\circ$ output of a globally fresh name (bound output).

The transition calculus is defined below. Note that some rules have (omitted) symmetric counterparts, while in all rules but ABS we have $P, P_1, P_2 \in \Pi_0$.

$$\begin{array}{c} \text{TAU} \frac{}{\tau.P \xrightarrow{\tau} P} \quad \text{OUT} \frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P} \quad \text{INP} \frac{}{x.\hat{P} \xrightarrow{xy} \hat{P}\mathcal{O}y} \quad y\#x\hat{P} \\ \text{INP} \frac{}{x.\hat{P} \xrightarrow{x\bullet} \hat{P}} \quad \text{SUM} \frac{P_1 \xrightarrow{a} Q}{P_1 + P_2 \xrightarrow{a} Q} \quad \text{SUM} \frac{P_1 \xrightarrow{x\bullet} \hat{Q}}{P_1 + P_2 \xrightarrow{xy} \hat{Q}\mathcal{O}y} \quad y\#P_1 \\ \text{ABS} \frac{P \xrightarrow{a} Q}{\langle x \rangle P \xrightarrow{\langle x \rangle a} \langle x \rangle Q} \quad \text{PAR} \frac{P_1 \xrightarrow{a} Q}{P_1 \mid P_2 \xrightarrow{a} Q \mid P_2} \quad \text{PAR} \frac{P_1 \xrightarrow{x\bullet} \hat{Q}}{P_1 \mid P_2 \xrightarrow{xy} \hat{Q}\mathcal{O}y \mid P_2} \quad y\#P_1 \\ \text{MATCH} \frac{P \xrightarrow{a} Q}{[x = x]P \xrightarrow{a} Q} \quad \text{RES} \frac{P \xrightarrow{a} Q}{\nu\langle y \rangle P \xrightarrow{a} \nu\langle y \rangle Q} \quad y\#a \quad \text{OPEN} \frac{P \xrightarrow{\bar{x}y} Q}{\nu\langle y \rangle P \xrightarrow{\bar{x}\circ} \langle y \rangle Q} \quad x\#y \\ \text{COMM} \frac{P_1 \xrightarrow{\bar{x}y} Q_1 \quad P_2 \xrightarrow{xy} Q_2}{P_1 \mid P_2 \xrightarrow{\tau} Q_1 \mid Q_2} \quad \text{COMM} \frac{P_1 \xrightarrow{\bar{x}y} Q_1 \quad P_2 \xrightarrow{x\bullet} \hat{Q}_2}{P_1 \mid P_2 \xrightarrow{\tau} Q_1 \mid \hat{Q}_2\mathcal{O}y} \quad y\#P_2 \\ \text{CLOSE} \frac{P_1 \xrightarrow{\bar{x}\circ} \hat{Q}_1 \quad P_2 \xrightarrow{x\bullet} \hat{Q}_2}{P_1 \mid P_2 \xrightarrow{\tau} \nu(\hat{Q}_1 \mid \hat{Q}_2)} \quad \text{REC} \frac{\langle \bar{x} \rangle P \mathcal{O}\bar{\alpha} \xrightarrow{a} Q}{p(\bar{\alpha}) \xrightarrow{a} Q} \quad p(\bar{x}) \triangleq P \end{array} \quad \Delta$$

We note here that the nominal π -calculus of [Gab03] is given in Π_0 (excluding recursion) and Act_0 , and the transition relation is a subset of $\Pi_0 \times \langle \mathbb{N} \rangle (Act_0 \times \Pi_0)$. The presentation we have given here, of $\rightarrow \subseteq \Pi \times Act \times \Pi$, is more convenient when considering strings of actions and algorithmic interpretations.

The transition relation defined above is early on known (non-fresh) names and late on fresh ones. That is, an input process $x.\hat{P}$ may either receive a known name y and move to $\hat{P}\mathcal{O}y$ (early), or receive a fresh name and move to \hat{P} (late). This allows for a finitely branching formulation of what is essentially an early transition semantics. *Note that the set of a process' known names is exactly its support*, in contrast to [BHLM04] where known names are explicitly inserted in the transition relation as a third argument.

Moreover, the transition relation is equivariant; in fact, it is preserved under any finite series of swappings and substitutions.

Lemma 14 For any $P, Q, \hat{Q} \in \Pi$, $a \in Act$ and $x, y \in \mathbb{N}$,

- if $P \xrightarrow{a} Q$ is derivable by \mathcal{D} , then $(x \ y) \cdot \mathcal{D}$ derives $(x \ y) \cdot P \xrightarrow{(x \ y) \cdot a} (x \ y) \cdot Q$,
- if $P \xrightarrow{x\bullet} \hat{Q}$ then, for every name $z \# P$, $P \xrightarrow{xz} \hat{Q}\mathcal{O}z$. \square

Proposition 15 Let $\vec{\sigma} = \sigma_1, \dots, \sigma_n$ and $\vec{\tau} = \tau_1, \dots, \tau_m$ be finite series of swappings and substitutions, $P, Q \in \Pi$ and $a \in \text{Act}$, then,

- $(\forall x \in \mathbf{S}(P). \vec{\sigma}(x) = \vec{\tau}(x)) \implies \vec{\sigma} \cdot P = \vec{\tau} \cdot P$,
- $P \xrightarrow{a} Q \implies \vec{\sigma} \cdot P \xrightarrow{\vec{\sigma} \cdot a} \vec{\sigma} \cdot Q$ □

We can define structural congruence for $\mathcal{N}\pi$, as follows.

Definition 16 A relation $\mathcal{R} \subseteq \Pi^2$ is called a congruence if,

- \mathcal{R} is reflexive, symmetric and transitive in Π_0 ,
- for any $P, Q, R \in \Pi_0$ and $x, y \in \mathbf{N}$, if $(P, Q) \in \mathcal{R}$ then \mathcal{R} also contains:
 $(P|R, Q|R), (P+R, Q+R), (\nu\langle y \rangle P, \nu\langle y \rangle Q), (x.\langle y \rangle P, x.\langle y \rangle Q), (\bar{x}y.P, \bar{x}y.Q), (\tau.P, \tau.Q)$
- for any $P, Q \in \Pi$ and $x \in \mathbf{N}$, if $(P, Q) \in \mathcal{R}$ then $(\langle x \rangle P, \langle x \rangle Q) \in \mathcal{R}$.

Structural congruence is the smallest congruence relation, \equiv , such that, for every $P, Q, R \in \Pi_0$ and every $x, y, z \in \mathbf{N}$ with $z \# P$, \equiv contains:

$$\begin{aligned} & (P|\mathbf{0}, P), (P|Q, Q|P), ((P|Q)|R, P|(Q|R)), (P+\mathbf{0}, P), (P+Q, Q+P) \\ & ((P+Q)+R, P+(Q+R)), (\nu\langle x \rangle \mathbf{0}, \mathbf{0}), (\nu\langle z \rangle (P|Q), (P|\nu\langle z \rangle Q)) \\ & (\nu\langle x \rangle \nu\langle y \rangle P, \nu\langle y \rangle \nu\langle x \rangle P), ([x = x]P, P), ([x = y]\mathbf{0}, \mathbf{0}) \end{aligned} \quad \Delta$$

In reasoning about the transition calculus, one traditionally considers some bisimilarity relation. Here we will consider a strong bisimilarity ([SW01, section 2.2]), i.e. one that takes equally into account τ and other actions. Now note that in ordinary π -calculus under early transition one has to tackle the problem of two processes being essentially bisimilar yet having different transitions because of ‘dummy’ names. For example, although the π -processes $\nu y.\bar{x}y.[z = z].\mathbf{0}$ and $\nu y.\bar{x}y.\mathbf{0}$ are essentially ‘the same’, the former cannot transit via $\bar{x}(z)$ as the latter can. In $\mathcal{N}\pi$ the analogous problem arises with inputs. For example, the essentially equivalent processes $x.\langle y \rangle [z = z]\mathbf{0}$ and $x.\langle y \rangle \mathbf{0}$ differ in that the latter cannot transit via xz . Such considerations lead to the following definition.

Definition 17 A relation $\mathcal{R} \subseteq \Pi^2$ is a bisimulation if it is symmetric and whenever $(P, Q) \in \mathcal{R}$ and $P \xrightarrow{a} P'$,

- if $a \notin \text{Act}_0$ then $\forall \alpha \in \mathbf{N}. (P@_\alpha, Q@_\alpha) \in \mathcal{R}$,
- else, if $a = x\bullet$ then there is Q' such that $Q \xrightarrow{x\bullet} Q'$ and $(P', Q') \in \mathcal{R}$, and, for all $y \in \mathbf{S}(Q) \setminus \mathbf{S}(P)$, there is Q_y such that $Q \xrightarrow{xy} Q_y$ and $(P'@_y, Q_y) \in \mathcal{R}$,
- else, if $a = xy$ with $y \# Q$ then there is Q' s.t. $Q \xrightarrow{x\bullet} Q'$ and $(P', Q'@_y) \in \mathcal{R}$,
- otherwise, there is Q' such that $Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathcal{R}$.

Bisimilarity, $\sim \subseteq \Pi^2$, is the largest bisimulation relation. □

It is easy to see now that $x\langle y \rangle.[z = z]\mathbf{0} \sim x\langle y \rangle.\mathbf{0}$. More generally, using translations from and to the ordinary π -calculus, we can show that $P \equiv Q$ implies $P \sim Q$. For now, we show the following.

Lemma 18 *Let $P \in \langle \mathbb{N} \rangle^n \Pi_0$ and $Q \in \langle \mathbb{N} \rangle^m \Pi_0$, with $m, n > 0$. Then, $P \sim Q \iff (\exists \alpha \in \mathbb{N}. P @ \alpha \sim Q @ \alpha)$. Moreover, if $P \sim Q$ and $P \xrightarrow{a} P'$, for some P', a , then $m = n$ and $Q \xrightarrow{b} Q'$, some Q', b . \square*

Proposition 19 (\sim equivariant and transitive)

- For any $P, Q \in \Pi$ and $\alpha, \beta \in \mathbb{N}$, $P \sim Q \implies (\alpha \beta) \cdot P \sim (\alpha \beta) \cdot Q$.
- For any $P, Q, R \in \Pi$, $P \sim R \sim Q \implies P \sim Q$. \square

3.1 Algorithmic interpretations, Automata

The transition calculus of Π is finitely branching due to the nominal treatment of fresh names. This is very convenient, yet we can obtain more. In particular, we notice that as a process evolves it is affixed with information on the fresh names received or created, by means of name-abstractions in the front. However, it may be the case that not all of these names appear in the process, since as a process evolves it may lose names. For example, the process $p(x) \triangleq \nu \langle y \rangle \bar{x}y.p(y)$ creates an infinite transition sequence:

$$\nu \langle y \rangle \bar{x}y.p(y) \xrightarrow{\bar{x}\circ} \langle y \rangle p(y) \xrightarrow{\epsilon} \langle y \rangle \nu \langle z \rangle \bar{y}z.p(z) \xrightarrow{\langle y \rangle \bar{y}\circ} \langle yz \rangle p(z) \xrightarrow{\epsilon} \langle yz \rangle \nu \langle w \rangle \bar{z}w.p(w) \xrightarrow{\langle yz \rangle \bar{z}\circ} \dots$$

where y is abstracted in $\langle yz \rangle p(z)$, although it does not appear in $\langle z \rangle p(z)$. If we devised a separate mechanism to keep track of the abstracted names in a transition sequence and kept only strictly-abstracted states, then the abstraction of y would be redundant in $\langle yz \rangle p(z)$ and we could dismiss it. Thus, the above transition sequence would become:

$$\nu \langle y \rangle \bar{x}y.p(y) \xrightarrow{\bar{x}\circ} \langle y \rangle p(y) \xrightarrow{\epsilon} \langle y \rangle \nu \langle z \rangle \bar{y}z.p(z) \xrightarrow{\langle y \rangle \bar{y}\circ} \langle z \rangle p(z) \xrightarrow{\epsilon} \langle z \rangle \nu \langle w \rangle \bar{z}w.p(w) \xrightarrow{\langle z \rangle \bar{z}\circ} \dots$$

or simply $p(x) \xrightarrow{\bar{x}\circ} \langle y \rangle p(y) \leftarrow \langle y \rangle \bar{y}\circ$.

In order to provide such a mechanism we will use proper name-lists, introduced in section 2.1. As we saw in that section, a state (a process) can be represented by a strictly-abstracted state and a proper name-list. Hence, the transition trees can be represented by trees with pairs of strictly-abstracted processes and proper name-lists as nodes, or, even better, by automata with strictly-abstracted states and an auxiliary memory device, the latter being a proper name-list.

We introduce nominal automata, or \mathcal{N} -automata for short. \mathcal{N} -automata are automata with access to an auxiliary memory device, in particular a proper name-list. At each control step the whole list may be accessed and the list may grow unboundedly large. The states of a \mathcal{N} -automaton have finite support, and the automaton evaluates strings of input which contain names, symbols for fresh names and other equivariant symbols. The auxiliary list stores the fresh names encountered: if while evaluating a string of input a symbol for fresh name is encountered, the \mathcal{N} -automaton adds a fresh entry in its list of names. This allows us to abstract states solely at active fresh names, so the states are kept as strict abstractions of the basic states.

Definition 20 (\mathcal{N} -automata) Fix a (possibly infinite) nominal set Sts_0 of basic states. A \mathcal{N} -automaton is a construction $\mathcal{A} := \langle \mathcal{S}, \Sigma_{eqt}, \delta, P_0, L_0 \rangle$ attached with a proper name-list L , such that:

- ◇ $\mathcal{S} \subseteq Sts^s (= \langle \mathbb{N} \rangle^* Sts_0)$ is the set of states, with $\mathfrak{S}(\mathcal{S}) = \mathfrak{S}(P_0)$ and all states in \mathcal{S} are accepting. The set of *possible configurations of \mathcal{A}* is the set $\mathcal{S} \boxtimes \mathbb{L} := (\mathcal{S} \times \mathbb{L}) \cap (Sts^s \boxtimes \mathbb{L})$ of all compatible state-list pairs. The *abstraction-sup of \mathcal{A}* , written $N_{\mathcal{A}}$, is the least ordinal containing all $n \in \omega$ with $\mathcal{S} \cap \langle \mathbb{N} \rangle^n Sts_0 \neq \emptyset$.
- ◇ P_0 is the initial state.
- ◇ Σ_{eqt} is the set of equivariant input symbols, which generates the set of input symbols $\Sigma := \langle \mathbb{N} \rangle^* \Sigma_0$, where $\Sigma_0 := \mathbb{N} \cup \{\bullet, \circ\} \cup \Sigma_{eqt}$.
“ \bullet ” [“ \circ ”] is the symbol denoting a locally [resp. globally] fresh name.
- ◇ L_0 is the initial configuration of L , with $\mathfrak{S}(L_0) = \emptyset$ and $(P_0, L_0) \in \mathcal{S} \boxtimes \mathbb{L}$.
- ◇ $\delta : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S} \times \Sigma \times \mathbf{D})$ is a function assigning to each state P a set of transition instructions.⁵ $(P, (Q, \mathbf{a}, \mathbf{d})) \in \delta$ is denoted by $P \xrightarrow[\mathbf{d}]{\mathbf{a}} Q$ and means that, whenever the automaton is in a configuration (P, L) and a symbol a is received, if $a = \mathbf{a}[L]$ then L is updated to $\text{upd}(\mathbf{a}, \mathbf{d}, L)$ and the control proceeds to state Q . Further definitions:

$$\mathbf{a} \in \Sigma := \mathbb{N} \cup \{\bullet, \circ\} \cup \Sigma_{eqt} \cup N_{\mathcal{A}}$$

$$\mathbf{d} \in \mathbf{D} := \{\text{del}(\vec{m}) \mid \vec{m} \in N_{\mathcal{A}}^* \wedge |\vec{m}| \in N_{\mathcal{A}} \wedge \vec{m} \text{ strictly increasing}\}$$

$$\text{upd}(\mathbf{a}, \mathbf{d}, L) := \text{d}(\text{add}_{\mathbf{a}}(L))$$

$$\text{add}_{\mathbf{a}} := \text{addfresh} \text{ if } \mathbf{a} \in \{\bullet, \circ\}, \text{ and } \text{add}_{\mathbf{a}} := \text{id} (= L \mapsto L) \text{ otherwise}$$

Thus, tags $\mathbf{a} \in \Sigma$ may be symbols from Σ_0 or L -references denoted by $i \in N_{\mathcal{A}}$, and encode a (possibly abstracted) input symbol. The latter is obtained using $\mathbf{a}[\cdot] : \mathbb{L} \rightarrow \Sigma$, defined by:

$$\mathbf{a}[L] := \begin{cases} \mathcal{N}\vec{\alpha} \cdot \langle \vec{\alpha} \rangle (L @ \vec{\alpha})_i & , \text{ if } \mathbf{a} = i \\ \mathcal{N}\vec{\alpha} \cdot \langle \vec{\alpha} \rangle \mathbf{a} & , \text{ otherwise} \end{cases}$$

We impose the following restrictions on δ : if $P \xrightarrow[\text{del}(\vec{m})]{\mathbf{a}} Q$ then,

Thus, the automaton starts from an initial state and list, and evaluates strings that may contain abstractions of equivariant symbols c , of known names x , or of \bullet, \circ . Given an input symbol a and a current configuration (P, L) the automaton either halts; or accepts the symbol, updates the list and moves to a next state. List update depends on the input symbol being a fresh-name symbol \bullet, \circ and on the $\text{del}(\vec{m})$ associated with the transition. Fresh-name symbols add a fresh entry to the list, and $\text{del}(\vec{m})$ applied to the list discards names with indices \vec{m} .

We define bisimilarity between configurations of \mathcal{N} -automata as follows.

Definition 21 Let \mathcal{A}, \mathcal{B} be \mathcal{N} -automata with sets of states $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{B}}$ respectively, and common input alphabet $\Sigma = \langle \mathbb{N} \rangle^* \Sigma_0$. Then, $\mathcal{R} \subseteq (\mathcal{S}_{\mathcal{A}} \boxtimes \mathbb{L}) \times (\mathcal{S}_{\mathcal{B}} \boxtimes \mathbb{L})$ is an \mathcal{AB} -bisimulation relation if, whenever $(P, L) \mathcal{R} (Q, K)$, if $P \xrightarrow[\mathbf{d}]{\mathbf{a}} P'$ then:

- ◇ if $\mathbf{a}[L] \notin \Sigma_0$ then $L, K \notin \mathbb{L}_0$ and $\mathcal{N}\alpha \in \mathbb{N}. (P, L @ \alpha) \mathcal{R} (Q, K @ \alpha)$,
- ◇ else, if $\mathbf{a} = \bullet$ then $Q \xrightarrow[\mathbf{d}]{\bullet} Q'$ with $(P', \text{upd}(\bullet, \mathbf{d}, L)) \mathcal{R} (Q', \text{upd}(\bullet, \mathbf{d}', K))$,
and, for all $y \in \mathfrak{S}(Q, K) \setminus \mathfrak{S}(P, L)$, we have $Q \xrightarrow[\mathbf{d}]{\mathbf{a}'} Q'$ with $\mathbf{a}'[K] = y$ and $(P', \text{d}(\text{add}(y, L))) \mathcal{R} (Q', \text{upd}(\mathbf{a}', \mathbf{d}', K))$,

⁵ Essentially, δ is a ‘curried form’ of a transition relation of type $\mathcal{S} \boxtimes \mathbb{L} \times \Sigma \rightarrow \mathcal{P}(\mathcal{S} \boxtimes \mathbb{L})$.

- ◇ else, if $\mathbf{a}[L] = \alpha \# Q, K$ then $Q \circ_{\mathbf{d}}^{\bullet} \rightarrow Q'$ with $(P', \text{upd}(\bullet, \mathbf{d}, L)) \mathcal{R}(Q', \mathbf{d}(\text{add}(\alpha, K)))$,
 - ◇ else $Q \circ_{\mathbf{d}'}^{\mathbf{a}'}$ with $\mathbf{a}[L] = \mathbf{a}'[K]$ and $(P', \text{upd}(\mathbf{a}, \mathbf{d}, L)) \mathcal{R}(Q', \text{upd}(\mathbf{a}', \mathbf{d}', K))$;
- and viceversa if $Q \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q'$.

\mathcal{AB} -bisimilarity, $\sim_{\mathcal{AB}} \subseteq (\mathcal{S}_{\mathcal{A}} \boxtimes \mathbb{L}) \times (\mathcal{S}_{\mathcal{B}} \boxtimes \mathbb{L})$, is the largest \mathcal{AB} -bisimulation. Moreover, \mathcal{A} and \mathcal{B} are bisimilar, $\mathcal{A} \sim_{\mathcal{A}} \mathcal{B}$, if $(P_0, L_0) \sim_{\mathcal{AB}} (Q_0, K_0)$, for the initial configurations (P_0, L_0) and (Q_0, K_0) of \mathcal{A} and \mathcal{B} respectively. \triangle

The intuition behind the formulations regarding locally-fresh-name transitions should be seen in the context of bisimulation in the $\mathcal{N}\pi$ -calculus. Thus, a locally-fresh-name transition \mathbf{a} from P must be matched by a similar transition from Q ; moreover, any transition from Q that accepts a name y fresh for P but not for Q must be matched by the transition \mathbf{a} from P .

we will now use \mathcal{N} -automata to model processes of the $\mathcal{N}\pi$ -calculus. Let the nominal set of strict $\mathcal{N}\pi$ -calculus processes be $\Pi^{\mathbf{s}} := \langle \mathbb{N} \rangle^* \Pi_0$.

Definition 22 (δ^{π} and $\mathcal{N}\pi$ -automata) Assume $\Sigma_{\text{eqt}} = \{\tau, \bar{\cdot}\}$ and $N_{\mathcal{A}} = \omega$, and define a (universal) $\mathcal{N}\pi$ -instruction function $\delta^{\pi} : \Pi^{\mathbf{s}} \rightarrow \mathcal{P}(\Pi^{\mathbf{s}} \times \Sigma \times \mathbf{D})$, with $(Q, \mathbf{a}, \mathbf{d}) \in \delta^{\pi}(P)$ written $P \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q$, by the following rules.

$$\begin{array}{c}
\text{TAU} \frac{}{\tau.P \circ_{\mathbf{d}}^{\tau} \rightarrow P} \quad \text{INP} \frac{}{x.\hat{P} \circ_{\mathbf{d}}^{xy} \rightarrow \hat{P} @ y} \quad y \# x.\hat{P} \quad \text{INP} \frac{}{x.\hat{P} \circ_{\mathbf{d}}^{x\bullet} \rightarrow \hat{P}} \quad \forall \alpha \in \mathbb{N}. \alpha \# \hat{P} @ \alpha \\
\text{OUT} \frac{}{\bar{x}y.P \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow P} \quad \text{MATCH} \frac{P \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q}{[x = x]P \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q} \quad \text{INP} \frac{}{x.\hat{P} \circ_{\text{del}(1)}^{x\bullet} \rightarrow \forall \alpha. \hat{P} @ \alpha} \quad \forall \alpha \in \mathbb{N}. \alpha \# \hat{P} @ \alpha \\
\text{SUM} \frac{P_1 \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q}{P_1 + P_2 \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q} \quad \text{SUM} \frac{P_1 \circ_{\mathbf{d}}^{x\bullet} \rightarrow \hat{Q}}{P_1 + P_2 \circ_{\mathbf{d}}^{xy} \rightarrow \hat{Q} @ y} \quad y \# P_1 \quad \wedge y \# P_2 \quad \text{SUM} \frac{P_1 \circ_{\text{del}(1)}^{x\bullet} \rightarrow Q}{P_1 + P_2 \circ_{\mathbf{d}}^{xy} \rightarrow Q} \quad y \# P_1 \quad \wedge y \# P_2 \\
\text{PAR} \frac{P_1 \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q}{P_1 \mid P_2 \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q \mid P_2} \quad \text{PAR} \frac{P_1 \circ_{\text{del}(1)}^{x\bullet} \rightarrow Q}{P_1 \mid P_2 \circ_{\mathbf{d}}^{xy} \rightarrow Q \mid P_2} \quad y \# P_1 \quad \wedge y \# P_2 \quad \text{PAR} \frac{P_1 \circ_{\mathbf{d}}^{x\bullet} \rightarrow \hat{Q}}{P_1 \mid P_2 \circ_{\mathbf{d}}^{xy} \rightarrow \hat{Q} @ y \mid P_2} \quad y \# P_1 \quad \wedge y \# P_2 \\
\text{RES} \frac{P \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q}{\nu \langle y \rangle P \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow \nu \langle y \rangle Q} \quad y \# \mathbf{a} \quad \text{OPEN} \frac{P \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow Q}{\nu \langle y \rangle P \circ_{\text{del}(1)}^{\bar{x}o} \rightarrow Q} \quad x \# y \quad \wedge y \# Q \quad \text{OPEN} \frac{P \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow Q}{\nu \langle y \rangle P \circ_{\mathbf{d}}^{\bar{x}o} \rightarrow \langle y \rangle Q} \quad x \# y \quad \wedge y \# Q \\
\text{REC} \frac{(\langle \bar{x} \rangle P) @ \bar{\alpha} \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q}{p(\bar{\alpha}) \circ_{\mathbf{d}}^{\mathbf{a}} \rightarrow Q} \quad p(\bar{x}) \triangleq P \quad \text{ABS} \frac{P \circ_{\text{del}(\bar{m})}^{\mathbf{a}} \rightarrow Q}{\langle x \rangle P \circ_{\text{del}(1, \bar{m}^{++})}^{\{1/x\} \cdot (\mathbf{a}^{++})} \rightarrow Q} \quad x \# P \quad \wedge x \# Q \quad \text{ABS} \frac{P \circ_{\text{del}(\bar{m})}^{\mathbf{a}} \rightarrow Q}{\langle x \rangle P \circ_{\text{del}(\bar{m}^{++})}^{\{1/x\} \cdot (\mathbf{a}^{++})} \rightarrow \langle x \rangle Q} \quad x \# P, Q \\
\text{COMM} \frac{P_1 \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow Q_1 \quad P_2 \circ_{\text{del}(1)}^{x\bullet} \rightarrow Q_2}{P_1 \mid P_2 \circ_{\mathbf{d}}^{\tau} \rightarrow Q_1 \mid Q_2} \quad \text{COMM} \frac{P_1 \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow Q_1 \quad P_2 \circ_{\mathbf{d}}^{x\bullet} \rightarrow \hat{Q}_2}{P_1 \mid P_2 \circ_{\mathbf{d}}^{\tau} \rightarrow Q_1 \mid \hat{Q}_2 @ y} \quad y \# \hat{Q}_2 \\
\text{COMM} \frac{P_1 \circ_{\mathbf{d}}^{\bar{x}y} \rightarrow Q_1 \quad P_2 \circ_{\mathbf{d}}^{xy} \rightarrow Q_2}{P_1 \mid P_2 \circ_{\mathbf{d}}^{\tau} \rightarrow Q_1 \mid Q_2} \quad \text{CLOSE} \frac{P_1 \circ_{\mathbf{d}}^{\bar{x}o} \rightarrow Q_1 \quad P_2 \circ_{\mathbf{d}}^{x\bullet} \rightarrow Q_2}{P_1 \mid P_2 \circ_{\mathbf{d}}^{\tau} \rightarrow \nu(Q_1 \mid Q_2)}
\end{array}$$

where each $P \circ_{\mathbf{d}}^{\mathbf{a}_1 \mathbf{a}_2 \mathbf{a}_3} \rightarrow Q$ is short for $P \circ_{\mathbf{d}}^{\mathbf{a}_1} \rightarrow P^* \circ_{\mathbf{d}}^{\mathbf{a}_2} \rightarrow P^{**} \circ_{\mathbf{d}}^{\mathbf{a}_3} \rightarrow Q$, using decompositions $\bar{x}y = (\bar{\cdot})(x)(y)$, etc., and dummy states P^*, P^{**} ; and also:

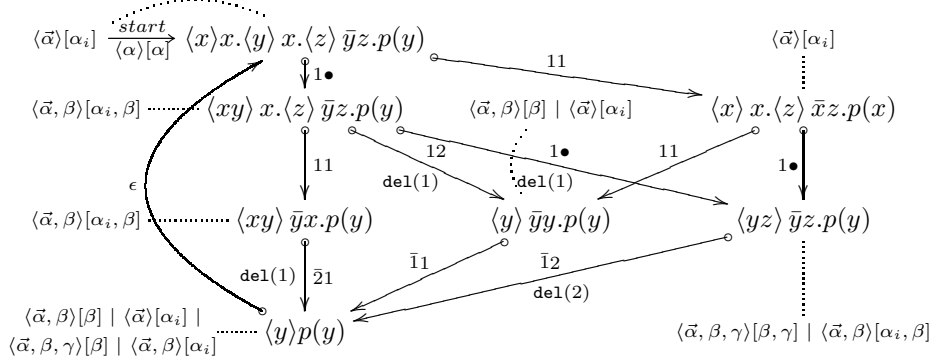
- ◇ In all rules but ABS we have $P, P_1, P_2 \in \Pi_0$ and missing \mathbf{d} -parts mean $\text{del}()$.

- ◇ The operator $_{}^{++}$ increases all integers in an expression by 1.
- ◇ In case $Q_1, Q_2 \in \Pi_0$ in CLOSE, we let $\nu(Q_1 | Q_2) := \mathcal{N}\alpha \cdot \nu\langle\alpha\rangle(Q_1 | Q_2)$.

To any $P_0 \in \Pi$ we relate a \mathcal{N} -automaton $\mathcal{A}_{P_0} := \langle \mathcal{S}, \{\tau, \bar{\cdot}\}, \delta, (P_0)^{\square}, (P_0)^{\square} \rangle$, where we set $\mathcal{S} := \{P \mid P \text{ is } \delta^\pi\text{-accessible from } (P_0)^{\square}\}$ and $\delta := \delta^\pi \upharpoonright \mathcal{S}$. We call \mathcal{A}_{P_0} a $\mathcal{N}\pi$ -automaton. \square

Note the utility of the two INP rules for fresh-name input: the first rule adds a fresh entry to L and abstracts on it; the second adds solely an empty abstraction to L (by adding a name, abstracting on it and the deleting it), since the received name does not appear in the rest of the process.

Let us proceed to an example: the $\mathcal{N}\pi$ -automaton for $\langle x \rangle x.\langle y \rangle x.\langle z \rangle \bar{y}z.p(y)$, with $p(x) \triangleq x.\langle y \rangle x.\langle z \rangle \bar{y}z.p(y)$, is depicted below. We write the initial configuration below the start arrow. Also, for ease of understanding, we attach to each state the corresponding configuration of its list, modulo recursion, and we use ϵ -transitions for recursion.



The succinctness of the $\mathcal{N}\pi$ -automata representation can be firstly seen in the following result. Let $\mathcal{N}\pi^{fc+\nu s}$ be the sub-calculus of $\mathcal{N}\pi$ containing those processes which have *finite control* and which reduce to processes whose ν -abstractions are *strict*⁶. The corresponding $\mathcal{N}\pi$ -automata have finitely many states.

Proposition 23 *If P_0 is in $\mathcal{N}\pi^{fc+\nu s}$ then \mathcal{A}_{P_0} is finite.* \square

It is possible to extend the above result to *finitary processes* in general, as it is done in the case of θ -automata. For this, one has to work in Π_0 modulo structural congruence \equiv . We do not pursue this here.

Finally, we show the intended correspondence between transitions in $\mathcal{N}\pi$ and δ^π , which allows us to show that bisimilarity between processes in $\mathcal{N}\pi$ is equivalent to bisimilarity of their $\mathcal{N}\pi$ -automata.

Lemma 24 *Let $P, Q \in \Pi^s$, $L \in \mathbb{L}$, $(a, d) \in \Sigma \times \mathbf{D}$. If $(P, L) \in \Pi^s \boxtimes \mathbb{L}$ and $P \circ \frac{a}{d} \rightarrow Q$ in δ^π , then $(P, L)^{\square} \xrightarrow{a[L]} (Q, \text{upd}(a, d, L))^{\square}$ in $\mathcal{N}\pi$.* \square

⁶ A process has finite control if no parallel compositions appear in its recursive definitions. $\nu \hat{P}$ is a strict ν -abstraction if $\mathcal{N}\alpha \in \mathbf{N}. \alpha \# \hat{P} @ \alpha$.

Lemma 25 For any $P, Q \in \Pi$, $a \in \text{Act}$ and $L \in \mathbb{L}$ with $(P, L) \in \Pi \boxtimes \mathbb{L}$ and $P \xrightarrow{a} Q$ in $\mathcal{V}\pi$, there exists $(\mathbf{a}, \mathbf{d}) \in \Sigma \times \mathbf{D}$ such that $\mathbf{a}[L] = a$ and $(P, L)^{\boxtimes} \xrightarrow{\mathbf{a}} (Q, \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxtimes}$ in δ^π . \square

Theorem 26 For any $P_0, Q_0 \in \Pi$, $P_0 \sim Q_0 \iff \mathcal{A}_{P_0} \sim_A \mathcal{A}_{Q_0}$. \square

4 Conclusion

This paper provides an investigation on the applicability of nominal theory to defining the π -calculus under early transition semantics. The resulting presentation deals elegantly with fresh-name introduction and produces a finitely-branching transition calculus. Moreover, it leads to considerations of nominal automata, \mathcal{V} -automata, for these transitions. \mathcal{V} -automata have a restricted state-space, as one state is needed for each fresh-name introduction step instead of one state for each possible fresh name. Each fresh-name introduction leads to a state with one more name-abstraction than the current state. To cope with the accumulation of irrelevant name-abstractions in states, a \mathcal{V} -automaton stores its history of name-abstractions in a proper name-list.

A major issue which has not been pursued in this paper is that of bisimulation-decidability for finite \mathcal{V} -automata; it is left for future consideration. So is the issue of (nominal) algorithms related to this class of automata.

References

- [BHLM04] Roberto Bruni, Furio Honsell, Marina Lenisa, and Marino Miculan. Modeling fresh names in the π -calculus using abstractions. *Electronic Notes in Theoretical Computer Science*, 106:25–41, 2004.
- [Gab00] M. J. Gabbay. *A Theory of Inductive Definitions with α -Equivalence: Semantics, Implementation, Programming Language*. DPhil thesis, University of Cambridge Computing Laboratory, 2000.
- [Gab03] M. J. Gabbay. The π -calculus in FM. In Fairouz Kamareddine, editor, *Thirty-five years of Automath*. Kluwer, 2003.
- [GP02] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.
- [HMS01] Furio Honsell, Marino Miculan, and Ivan Scagnetto. π -calculus in (co)inductive-type theory. *Theor. Comput. Science*, 253(2):239–285, 2001.
- [MP99] Dale Miller and Catuscia Palmidessi. Foundational aspects of syntax. *ACM Comput. Surv.*, 31(3es):11, 1999.
- [MP05] Ugo Montanari and Marco Pistore. Structured coalgebras and minimal HD-automata for the π -calculus. *Theor. Comput. Science*, 340(3):539–576, 2005.
- [Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.
- [SW01] Davide Sangiorgi and David Walker. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

A Appendix: Deferred Proofs

Proof of proposition 3:

Note first that if such an \tilde{f} exists, then $\forall x \in X. \forall \alpha \in \mathbf{N}. \alpha \# f(\alpha, x)$, since for any $x \in X$ and any $\alpha \# x$ such that $\tilde{f}(x) = f(\alpha, x)$, if $\beta \# x$, $\tilde{f}(x)$,

$$\alpha = (\alpha \beta) \cdot \beta \# (\alpha \beta) \cdot \tilde{f}(x) = \tilde{f}(x) = f(\alpha, x)$$

Conversely, suppose that that $\forall x \in X. \forall \alpha \in \mathbf{N}. \alpha \# f(\alpha, x)$. Define

$$\tilde{f} := \{(x, y) \mid \forall \alpha \in \mathbf{N}. \alpha \# f(\alpha, x) \wedge y = f(\alpha, x)\}$$

We show first that \tilde{f} is a partial function: if $(x, y), (x, z) \in \tilde{f}$, then for cofinitely many α we have $\alpha \# f(\alpha, x) = y$, and for cofinitely many α we have $\alpha \# f(\alpha, x) = z$. Hence, there are cofinitely many α that do both:

$$\alpha \# f(\alpha, x) = y = z$$

Moreover, \tilde{f} is total. Take any $x \in X$, then there is a cofinite set $A \subseteq \mathbf{N}$ such that $\forall \alpha \in A. \alpha \# f(\alpha, x)$. Take $\alpha \neq \beta \in A$ with $\alpha, \beta \# x$ (so $\beta \# f(\alpha, x)$), then

$$f(\beta, x) = f((\alpha \beta) \cdot \alpha, (\alpha \beta) \cdot x) = (\alpha \beta) \cdot f(\alpha, x) = f(\alpha, x)$$

Hence, there exists $y \in Y$ such that $\forall \alpha \in A \setminus \mathbf{S}(x). \alpha \# f(\alpha, x) = y$. Since $A \setminus \mathbf{S}(x)$ is cofinite, $\tilde{f}(x) = y$.

Moreover, $\tilde{f} : X \rightarrow Y$ is equivariant:

$$\begin{aligned} (\alpha \beta) \cdot \tilde{f}(x) = y &\iff \tilde{f}(x) = (\alpha \beta) \cdot y \\ &\iff \forall \gamma \in \mathbf{N}. \gamma \# f(\gamma, x) \wedge (\alpha \beta) \cdot y = f(\gamma, x) \\ &\iff \forall \gamma \in \mathbf{N}. (\alpha \beta) \cdot \gamma \# (\alpha \beta) \cdot f(\gamma, x) \wedge y = (\alpha \beta) \cdot f(\gamma, x) \\ &\stackrel{(*)}{\iff} \forall \gamma \in \mathbf{N}. \gamma \# f(\gamma, (\alpha \beta) \cdot x) \wedge y = f(\gamma, (\alpha \beta) \cdot x) \iff \tilde{f}((\alpha \beta) \cdot x) = y \end{aligned}$$

In step (*) note that a sentence holds for cofinitely many $\gamma \in \mathbf{N}$ iff it holds for cofinitely many such with $\gamma \# \alpha, \beta$. For similar reasons,

$$\tilde{f} = \{(x, y) \mid \forall \alpha \in \mathbf{N}. y = f(\alpha, x)\} \quad \square$$

Proof of proposition 5:

Take any $x \in X, y \in Y$. Let $z_x = \forall \alpha \cdot g(\alpha, x)$ be the cofinitely constant value of $g(_, x)$, and $w_{y,x} = \forall \beta \cdot f(\beta, y, z_x)$ the cofinitely constant value of $f(_, y, z_x)$. Then, there are cofinite sets $A_x \subseteq \mathbf{N}, B_{y,x} \subseteq \mathbf{N}$ such that $\forall \alpha \in A_x. g(\alpha, x) = z_x$ and $\forall \beta \in B_{y,x}. f(\beta, y, z_x) = w_{y,x}$. Hence, $\forall \beta \in A_x \cap B_{y,x}. z_x = g(\beta, x) \wedge w_{y,x} = f(\beta, y, z_x)$, as required. \square

Proof of proposition 10:

Note first that, for any $(P, L) \in \text{Sts}^s \boxtimes \mathbb{L}$, if $L = \text{push}(\alpha, L')$, some α, L' , then $(P, L)^\square = (P @ \alpha, L')^\square$. This is shown by induction on L : for $L \in \mathbb{L}_0$ it is

clear, otherwise, $(P, L)^\square = \mathcal{V}\beta \cdot \langle \beta \rangle (P, L @ \beta)^\square = \mathcal{V}\beta \cdot \langle \beta \rangle (P, \text{push}(\alpha, L' @ \beta))^\square \stackrel{IH}{=} \mathcal{V}\beta \cdot \langle \beta \rangle (P @ \alpha, L' @ \beta)^\square = (P @ \alpha, L')^\square$.

Now, for the first statement let $L \in \langle \mathbf{N} \rangle^n \mathbb{L}_0$; we proceed by induction on $n + |L|$. For the base case of $L = [\vec{\alpha}] \in \mathbb{L}_0$, we have $((P, L)^\square, L)^\square = (\langle \vec{\alpha} \rangle P, [\vec{\alpha}])^\square = P$. Now, suppose $\mathcal{V}\alpha \in \mathbf{N}$. $\alpha \# L @ \alpha$, then,

$$\begin{aligned} ((P, L)^\square, L)^\square &= (\mathcal{V}\alpha \cdot (P @ \alpha, L @ \alpha)^\square, L)^\square \\ &= \mathcal{V}\beta \cdot \langle \beta \rangle (\mathcal{V}\alpha \cdot (P @ \alpha, L @ \alpha)^\square, L @ \beta)^\square \\ &= \mathcal{V}\beta \cdot \langle \beta \rangle ((P @ \beta, L @ \beta)^\square, L @ \beta)^\square \stackrel{IH}{=} \mathcal{V}\beta \cdot \langle \beta \rangle (P @ \beta) = P \end{aligned}$$

On the other hand, if $\mathcal{V}\alpha \in \mathbf{N}$. $L @ \alpha = \text{push}(\alpha, L')$,

$$\begin{aligned} ((P, L)^\square, L)^\square &= (\mathcal{V}\beta \cdot \langle \beta \rangle (P @ \beta, L')^\square, L)^\square \\ &= \mathcal{V}\beta \cdot \langle \beta \rangle (\mathcal{V}\alpha \cdot \langle \alpha \rangle (P @ \alpha, L')^\square, L @ \beta)^\square \\ &\stackrel{\text{note}}{=} \mathcal{V}\beta \cdot \langle \beta \rangle ((P @ \beta, L')^\square, L')^\square \stackrel{IH}{=} \mathcal{V}\beta \cdot \langle \beta \rangle (P @ \beta) = P \end{aligned}$$

Finally, if $\mathcal{V}\alpha \in \mathbf{N}$. $\alpha \# L @ \alpha \wedge L = \text{push}(\beta, L')$, then,

$$((P, L)^\square, L)^\square = (\langle \beta \rangle (P, L')^\square, L)^\square \stackrel{\text{note}}{=} ((P, L')^\square, L')^\square \stackrel{IH}{=} P$$

For the second statement we proceed dually by induction on $n + |L|$. For the base case of $L = [\vec{\alpha}] \in \mathbb{L}_0$ we have,

$$((Q, L)^\square, L)^\square = (Q @ \vec{\alpha}, [\vec{\alpha}])^\square = Q$$

Now, suppose $\mathcal{V}\alpha \in \mathbf{N}$. $\alpha \# L @ \alpha$, then,

$$\begin{aligned} ((Q, L)^\square, L)^\square &= (\mathcal{V}\alpha \cdot \langle \alpha \rangle (Q, L @ \alpha)^\square, L)^\square \\ &= \mathcal{V}\beta \cdot ((\mathcal{V}\alpha \cdot \langle \alpha \rangle (Q, L @ \alpha)^\square) @ \beta, L @ \beta)^\square \\ &= \mathcal{V}\beta \cdot ((Q, L @ \beta)^\square, L @ \beta)^\square \stackrel{IH}{=} Q \end{aligned}$$

On the other hand, if $\mathcal{V}\alpha \in \mathbf{N}$. $L @ \alpha = \text{push}(\alpha, L')$, then,

$$\begin{aligned} ((Q, L)^\square, L)^\square &= (\mathcal{V}\alpha \cdot \langle \alpha \rangle (Q, L @ \alpha)^\square, L)^\square \stackrel{\text{note}}{=} (\mathcal{V}\alpha \cdot \langle \alpha \rangle (Q @ \alpha, L')^\square, L)^\square \\ &= \mathcal{V}\beta \cdot (\langle \beta \rangle ((\mathcal{V}\alpha \cdot \langle \alpha \rangle (Q @ \alpha, L')^\square) @ \beta, L')^\square) \\ &= \mathcal{V}\beta \cdot (\langle \beta \rangle ((Q @ \beta, L')^\square, L')^\square) \stackrel{IH}{=} \mathcal{V}\beta \cdot \langle \beta \rangle (Q @ \beta) = Q \end{aligned}$$

Finally, if $\mathcal{V}\alpha \in \mathbf{N}$. $\alpha \# L @ \alpha \wedge L = \text{push}(\beta, L')$, then,

$$((Q, L)^\square, L)^\square \stackrel{\text{note}}{=} ((Q @ \beta, L')^\square, L)^\square = \langle \beta \rangle ((Q @ \beta, L'), L') \stackrel{IH}{=} \langle \beta \rangle (Q @ \beta) = Q \quad \square$$

Proof of proposition 12:

The first conjunct of the first statement is proven by straightforward induction

on P . For the second conjunct we have that

$$y \# P \implies y \in \mathcal{S}(P) \implies \sigma \cdot y \in \sigma \cdot P \implies \sigma \cdot y \# \sigma \cdot P .$$

Now, for the second statement, denote for economy $\sigma \cdot y$ by y' and $\sigma \cdot \hat{P}$ by \hat{P}' .

$$\text{If } y \# \hat{P}, \text{ then } \sigma \cdot (\hat{P} \mathcal{Q} y) = \sigma \cdot (\hat{P} @ y) = \mathcal{N} \alpha \cdot \sigma \cdot (y \alpha) \cdot (\hat{P} @ \alpha) .$$

Moreover, if $y' \# \hat{P}'$ then,

$$\hat{P}' \mathcal{Q} y' = \mathcal{N} \alpha \cdot \{y' / \alpha\} (\sigma \hat{P}') @ \alpha = \mathcal{N} \alpha \cdot \{y' / \alpha\} (\mathcal{N} \beta \cdot (\beta) \sigma (\hat{P} @ \beta)) @ \alpha = \mathcal{N} \alpha \cdot \{y' / \alpha\} \sigma (\hat{P} @ \alpha)$$

We claim that, for any such $\alpha \# y, \sigma$ and any $x \in \mathcal{S}(\hat{P} @ \alpha)$,

$$\begin{aligned} \{y' / \alpha\} \cdot \sigma \cdot x &= \sigma \cdot (y \alpha) \cdot x \quad \text{and hence} \\ \sigma \cdot (\hat{P} \mathcal{Q} y) &= \sigma \cdot (\mathcal{N} \alpha \cdot (y \alpha) \cdot (\hat{P} @ \alpha)) = \mathcal{N} \alpha \cdot \sigma \cdot (y \alpha) \cdot (\hat{P} @ \alpha) = \hat{P}' \mathcal{Q} y' \end{aligned}$$

Indeed, if $x = \alpha$ then $\sigma \cdot x = x = \alpha$, so $\{y' / \alpha\} \cdot \sigma \cdot x = \{y' / \alpha\} \cdot \alpha = y' = \sigma \cdot (y \alpha) \cdot x$.

If $\alpha \# x$ then also $\alpha \# \sigma \cdot x$, so $\{y' / \alpha\} \cdot \sigma \cdot x = \sigma \cdot x \stackrel{y \# \hat{P} \Rightarrow y \# x}{=} \sigma \cdot (y \alpha) \cdot x$.

Otherwise, if $y' \# \hat{P}'$ then,

$$\hat{P}' \mathcal{Q} y' = \hat{P}' @ y' = (\mathcal{N} \alpha \cdot \langle \alpha \rangle \sigma \cdot (\hat{P} @ \alpha)) @ y' = \mathcal{N} \alpha \cdot (y' \alpha) \cdot \sigma \cdot (\hat{P} @ \alpha)$$

and note that, for $\alpha \# \sigma, y$, functions $(y' \alpha) \circ \sigma$ and $\sigma \circ (y \alpha)$ are equal (in \mathbf{N}).

Now, if $y \# \hat{P}$ then $y' \# \hat{P}'$, by the previous lemma. Therefore,

$$\hat{P}' \mathcal{Q} y' = \mathcal{N} \alpha \cdot \{y' / \alpha\} \cdot \sigma \cdot (\hat{P} @ \alpha) \quad \text{and} \quad \sigma \cdot (\hat{P} \mathcal{Q} y) = \mathcal{N} \alpha \cdot \sigma \cdot \{y / \alpha\} \cdot (\hat{P} @ \alpha)$$

and now note, for $\alpha \# \sigma, y$, the functions $\{y' / \alpha\} \circ \sigma$ and $\sigma \circ \{y / \alpha\}$ are equal. \square

Proof of proposition 15:

The former property is shown by induction on P , using the fact that $\vec{\sigma} \cdot x = \vec{\sigma}(x)$ for every name x .

For the latter, assume $P \xrightarrow{a} Q$ and take $\vec{\sigma}$ a finite series of substitutions and swappings, say $\vec{\sigma} := \sigma_1, \dots, \sigma_n$. We argue by induction on n ; for $n = 0$ we have that $\vec{\sigma}$ acts on P, a, Q like the identity permutation, because of the former property.

Now suppose $n > 0$; we have $\vec{\sigma} = \sigma, \vec{\tau}$, for some series $\vec{\tau}$ of length $n - 1$, so, by induction hypothesis, $P' \xrightarrow{a'} Q'$ holds for $P' := \vec{\tau} \cdot P, Q' := \vec{\tau} \cdot Q$ and $a' := \vec{\tau} \cdot a$.

We need to show that $\sigma \cdot P' \xrightarrow{\sigma \cdot a'} \sigma \cdot Q'$ also holds. If σ is a name-swapping use the previous lemma.

Otherwise, if $\sigma = \{x_1 / x_2\}$ for $x_1, x_2 \in \mathbf{N}$, then we (again) proceed by induction on the derivation of $P' \xrightarrow{a'} Q'$, using lemmata 12 and 14, and manipulations of $\mathcal{N} \alpha$'s. Below we show two characteristic examples of the induction step.

Suppose $P' \xrightarrow{a'} Q'$ is in fact $\nu \langle y \rangle P \xrightarrow{\bar{x} \circ} \langle y \rangle Q$, derived by $P \xrightarrow{xy} Q$ with $y \# x$.

By lemma 14, we can derive $(\alpha y) \cdot P \xrightarrow{\bar{x} \alpha} (\alpha y) \cdot Q$, for some fresh α , in the same number of steps. Thus, the IH is applicable, so $\sigma \cdot (\alpha y) \cdot P \xrightarrow{\sigma \cdot \bar{x} \alpha} \sigma \cdot (\alpha y) \cdot Q$ is derivable. Applying OPEN, we get $\nu \langle \alpha \rangle \sigma \cdot (\alpha y) \cdot P \xrightarrow{\sigma \cdot \bar{x} \alpha} \langle \alpha \rangle \sigma \cdot (\alpha y) \cdot Q$.

Suppose $P' \xrightarrow{a'} Q'$ is in fact $p(\bar{\alpha}) \xrightarrow{a'} Q'$, derived by $(\langle \bar{x} \rangle P) @ \bar{\alpha} \xrightarrow{a'} Q'$. By

IH, $\sigma \cdot (\langle \vec{x} \rangle P) \mathcal{O} \vec{\alpha} \xrightarrow{\sigma \cdot a'} \sigma \cdot Q'$. Now note that, by lemma 12, $\sigma \cdot (\langle \vec{x} \rangle P) \mathcal{O} \sigma \cdot \vec{\alpha} = (\sigma \cdot \langle \vec{x} \rangle P) \mathcal{O} \sigma \cdot \vec{\alpha}$. But $\langle \vec{x} \rangle P$ is equivariant, so the action of σ on it is the same as the action of the identity permutation. Hence, $(\langle \vec{x} \rangle P) \mathcal{O} \sigma \cdot \vec{\alpha} \xrightarrow{\sigma \cdot a'} \sigma \cdot Q'$, and thus, by REC, $p(\sigma \cdot \vec{\alpha}) \xrightarrow{\sigma \cdot a'} \sigma \cdot Q'$, as required. \square

Proof of lemma 18:

The first statement is straightforward, by definition of bisimilarity. For the second, suppose $P \xrightarrow{a} P'$. Then, $\forall \alpha \in \mathbf{N}. P @ \alpha \xrightarrow{a @ \alpha} P' @ \alpha \wedge P @ \alpha \sim Q @ \alpha$, and repeating this n times we get $\forall \vec{\alpha} \in \mathbf{N}. P @ \vec{\alpha} \xrightarrow{a @ \vec{\alpha}} P' @ \vec{\alpha} \wedge P @ \vec{\alpha} \sim Q @ \vec{\alpha}$, where $|\vec{\alpha}| = n$. Now $Q @ \vec{\alpha}$ has to simulate $a @ \vec{\alpha} \in Act_0$, hence $Q @ \vec{\alpha} \in \Pi_0$, $\therefore m = n$, and $Q @ \vec{\alpha} \xrightarrow{a''} Q''$. Applying ABS n times we get $Q \xrightarrow{\langle \vec{\alpha} \rangle a''} \langle \vec{\alpha} \rangle Q''$. \square

Proof of proposition 19:

For the first statement, take $\mathcal{R} := \{(P, Q) \mid (\alpha \beta) \cdot P \sim (\alpha \beta) \cdot Q\}$. It is straightforward to show that \mathcal{R} is a bisimulation.

For the second statement, take $\mathcal{R} := \{(P, Q) \mid \exists R \in \Pi. P \sim R \sim Q\}$. We show that \mathcal{R} is a bisimulation. Note first that \mathcal{R} is symmetric. Moreover, suppose that $(P, Q) \in \mathcal{R}$, so $P \sim R \sim Q$, and $P \xrightarrow{a} P'$.

If $a \notin Act_0$ then $\forall \alpha \in \mathbf{N}. P @ \alpha \sim R @ \alpha$ and $R \xrightarrow{a'} R'$, some R' and $a' \notin Act_0$. Thus $\forall \alpha \in \mathbf{N}. R @ \alpha \sim Q @ \alpha$, hence $\forall \alpha \in \mathbf{N}. P @ \alpha \sim R @ \alpha \sim Q @ \alpha$, $\therefore \forall \alpha \in \mathbf{N}. (P @ \alpha, Q @ \alpha) \in \mathcal{R}$.

Else, if $a = x \bullet$ then $R \xrightarrow{x \bullet} R' \sim P'$, $\therefore Q \xrightarrow{x \bullet} Q' \sim R'$. Hence, $(P', Q') \in \mathcal{R}$. Moreover, if $y \in \mathbf{S}(Q) \setminus \mathbf{S}(P)$ and $y \# R$ then $Q \xrightarrow{xy} Q_y \sim R' @ y$. But, by the previous lemma and equivariance of \sim , $R' \sim P'$ implies $R' @ y \sim P' @ y$, thus $(P' @ y, Q_y) \in \mathcal{R}$. Otherwise, if $y \in \mathbf{S}(Q) \setminus \mathbf{S}(P)$ and $y \# R$ then $R \xrightarrow{xy} R_y \sim P' @ y$ and thus $Q \xrightarrow{xy} Q_y \sim R_y$, hence $(P' @ y, Q_y) \in \mathcal{R}$.

Else, if $a = xz$, some $z \# Q, R$, then $R \xrightarrow{x \bullet} R'$ with $R' @ z \sim P'$, $\therefore Q \xrightarrow{x \bullet} Q' \sim R'$ and $(P', Q' @ z) \in \mathcal{R}$. If $a = xz$, some $z \# Q$ but $z \# R$, then $R \xrightarrow{xz} R' \sim P'$, $\therefore Q \xrightarrow{x \bullet} Q'$ with $R' \sim Q' @ z$, hence $(P', Q' @ z) \in \mathcal{R}$.

Else, if $a = xz$, some $z \# R$ but $z \# Q$, then $R \xrightarrow{x \bullet} R'$ with $P' \sim R' @ z$, $\therefore Q \xrightarrow{xz} Q_z \sim R' @ z$, hence $(P', Q_z) \in \mathcal{R}$. The other cases are clear. \square

Proof of proposition 23:

Let $\mathcal{A}_{P_0} = \langle \mathcal{S}, \{\tau, \bar{\cdot}\}, \delta, P'_0, L_0 \rangle$. Moreover, suppose that P_0 invokes the definitions $p_i(\vec{z}_i) \triangleq P_i$, $i = 1, \dots, N$, and take $M := |P'_0| + |P'_0| \cdot \max\{|P_i| \mid i \leq N\}$. The size function $|\cdot| : \Pi \rightarrow \omega$ is given by:

$$\begin{aligned} |\mathbf{0}| &:= 1, \quad |p(\vec{x})| := |\vec{x}|, \quad |\bar{x}y.P| := 2 + |P|, \quad |x.\hat{P}| := 1 + |\hat{P}|, \quad |P|Q := |P| + |Q|, \\ |P+Q| &:= |P| + |Q|, \quad |\nu\hat{P}| := |\hat{P}|, \quad |[x=y]P| := 2 + |P|, \quad |\hat{P}| := \forall \alpha \bullet |\hat{P} @ \alpha| \end{aligned}$$

Now, if Q is δ^π -accessible from P'_0 then $|Q| \leq M$, as P'_0 may only increase its size by recursion, and recursions cannot obtain size greater than $\max\{|P_i| \mid i \leq N\}$, as P_0 has finite control. Moreover, if $Q \in \langle \mathbf{N} \rangle^n \Pi_0$ then $n \leq |\vec{\alpha}| \bullet |Q @ \vec{\alpha}| = |Q|$.

Hence,

$$\mathcal{S} \leq \{Q \in \bigcup_{i=0}^M (\mathbb{N})^i \Pi_0 \mid |Q| \leq M \wedge \mathfrak{s}(Q) \subseteq \mathfrak{s}(P'_0)\}$$

Because of strict ν -abstractions, the RHS-set above is finite, so \mathcal{S} is finite. \square

Proof of lemma 24:

By induction on the derivation of $P \circ_{\mathfrak{a}}^{\mathfrak{a}} Q$. Note that $P \in \Pi_0$ implies $|L| = 0$, and thus $L = \langle \vec{\alpha} \rangle$, for any $\vec{\alpha} \in \mathbb{N}^\#$. In this case, $(P, L)^\square = \langle \vec{\alpha} \rangle P$ and $\mathfrak{a}[L] = \langle \vec{\alpha} \rangle \mathfrak{a}$, by choosing any sufficiently fresh $\vec{\alpha}$. We proceed with a case analysis on the last rule in the derivation and show some characteristic cases:

- ▶ If $\bar{x}y.Q \circ_{\mathfrak{a}}^{\bar{x}y} Q$, then $\langle \vec{\alpha} \rangle P \xrightarrow{\langle \vec{\alpha} \rangle \bar{x}y} \langle \vec{\alpha} \rangle Q$. TAU rule similarly.
- ▶ If $x.\hat{Q} \circ_{\mathfrak{a}}^{x\bullet} \hat{Q}$, then $\langle \vec{\alpha} \rangle P \xrightarrow{\langle \vec{\alpha} \rangle x\bullet} \langle \vec{\alpha} \rangle \hat{Q}$. Now, $(\hat{Q}, \mathfrak{add}(\langle \vec{\alpha} \rangle))^\square = (\hat{Q}, \mathfrak{V}\beta.\langle \vec{\alpha} \rangle \beta)^\square = \langle \vec{\alpha} \rangle \mathfrak{V}\beta.(\hat{Q}, \langle \beta \rangle \beta)^\square = \langle \vec{\alpha} \rangle \mathfrak{V}\gamma.\langle \gamma \rangle (\hat{Q} @ \gamma) = \langle \vec{\alpha} \rangle \hat{Q}$.
- ▶ If $x.\hat{Q} \circ_{\mathfrak{a}}^{\mathfrak{del}(1)} \mathfrak{V}\gamma.\hat{Q} @ \gamma$, then $\langle \vec{\alpha} \rangle P \xrightarrow{\langle \vec{\alpha} \rangle x\bullet} \langle \vec{\alpha} \rangle \hat{Q}$. Now, $(\mathfrak{V}\gamma.\hat{Q} @ \gamma, \mathfrak{del}(1)(\mathfrak{add}(\langle \vec{\alpha} \rangle)))^\square = (\mathfrak{V}\gamma.\hat{Q} @ \gamma, \mathfrak{V}\gamma.\langle \vec{\alpha} \rangle \gamma)^\square = \langle \vec{\alpha} \rangle (\hat{Q} @ \gamma) = \langle \vec{\alpha} \rangle \hat{Q}$.
- ▶ If $\frac{P_1 \circ_{\mathfrak{a}}^{x\bullet} \hat{Q}}{P_1 \mid P_2 \circ_{\mathfrak{a}}^{xy} \hat{Q} @ y \mid P_2} y \# P_1 \wedge y \# P_2$, then, by IH, $\langle \vec{\alpha} \rangle P_1 \xrightarrow{\langle \vec{\alpha} \rangle x\bullet} \langle \vec{\alpha} \rangle \hat{Q}$ and thus $P_1 \xrightarrow{x\bullet} \hat{Q}$. By PAR, $P_1 \mid P_2 \xrightarrow{xy} \hat{Q} @ y \mid P_2$, hence $\langle \vec{\alpha} \rangle (P_1 \mid P_2) \xrightarrow{\langle \vec{\alpha} \rangle xy} \langle \vec{\alpha} \rangle (\hat{Q} @ y \mid P_2)$.
- ▶ If $\frac{P_1 \circ_{\mathfrak{a}}^{x\bullet} \hat{Q}}{P_1 \mid P_2 \circ_{\mathfrak{a}}^{x\bullet} \hat{Q} \mid P_2}$, then, by IH, $\langle \vec{\alpha} \rangle P_1 \xrightarrow{\langle \vec{\alpha} \rangle x\bullet} \langle \vec{\alpha} \rangle \hat{Q}$ and thus $P_1 \xrightarrow{x\bullet} \hat{Q}$. By PAR, $P_1 \mid P_2 \xrightarrow{x\bullet} \hat{Q} \mid P_2 = \mathfrak{V}\beta.\langle \beta \rangle (\hat{Q} @ \beta \mid P_2)$, hence $\langle \vec{\alpha} \rangle (P_1 \mid P_2) \xrightarrow{\langle \vec{\alpha} \rangle x\bullet} \mathfrak{V}\beta.\langle \vec{\alpha} \rangle \beta (\hat{Q} @ \beta \mid P_2)$. Now, $(\hat{Q} \mid P_2, \mathfrak{V}\beta.\langle \vec{\alpha} \rangle \beta)^\square = \langle \vec{\alpha} \rangle \mathfrak{V}\beta.(\hat{Q} \mid P_2, \langle \beta \rangle \beta)^\square = \langle \vec{\alpha} \rangle \mathfrak{V}\gamma.\langle \gamma \rangle ((\hat{Q} \mid P_2) @ \gamma) = \langle \vec{\alpha} \rangle (\hat{Q} \mid P_2)$.
- ▶ If $\frac{P' \circ_{\mathfrak{a}}^{\mathfrak{del}(\vec{m})} Q'}{\langle x \rangle P' \circ_{\mathfrak{a}'}^{\{1/x\} \cdot (\mathfrak{a}^{++})} \langle x \rangle Q'} x \# P$, then $(\langle x \rangle P', L) \in \Pi^s \boxtimes \mathbb{L}$ implies that

either $L = \mathfrak{push}(\alpha, L')$ for some $\alpha \# P', L'$, or $L = \mathfrak{V}\alpha.\langle \alpha \rangle \mathfrak{push}(\alpha, L')$. In the former case we have that, $(\langle x \rangle P', L)^\square = ((\alpha x) \cdot P', L')^\square$, while in the latter,

$$(\langle x \rangle P', L)^\square = \mathfrak{V}\beta.\langle \beta \rangle (\langle x \rangle P', L @ \beta)^\square = \mathfrak{V}\beta.\langle \beta \rangle (\langle x \rangle P', \mathfrak{push}(\beta, L'))^\square = \mathfrak{V}\beta.\langle \beta \rangle ((\beta x) P', L')^\square$$

We treat only the latter case, the former is treated similarly. Take any $\beta \# x, P', L'$ and let $L'' := (\beta x) L'$. By IH we have that, $(P', L'')^\square \xrightarrow{\mathfrak{a}[L'']} (Q', \mathfrak{upd}(\mathfrak{a}, \mathfrak{d}, L''))^\square$, and hence, by equivariance of transition and of all used functions,

$$\text{Abs} \frac{((\beta x) \cdot P', L')^\square \xrightarrow{\langle \beta \rangle \mathfrak{a}[L'']} ((\beta x) \cdot Q', \mathfrak{upd}(\mathfrak{a}, \mathfrak{d}, L''))^\square}{\langle \beta \rangle ((\beta x) \cdot P', L')^\square \xrightarrow{\langle \beta \rangle \mathfrak{a}[L'']} \langle \beta \rangle ((\beta x) \cdot Q', \mathfrak{upd}(\mathfrak{a}, \mathfrak{d}, L''))^\square}$$

Now note that $\{1/x\} \cdot (\mathfrak{a}^{++})(L) = \mathfrak{V}\beta.\langle \beta \rangle ((\beta x) \cdot \mathfrak{a})(L')$ and $(\langle x \rangle Q', \mathfrak{upd}(\mathfrak{a}, \mathfrak{d}', L))^\square = \langle \beta \rangle ((\beta x) \cdot Q', \mathfrak{upd}(\mathfrak{a}, \mathfrak{d}, L'))^\square$. \square

Proof of lemma 25:

We formulate a translation from transitions in $\mathcal{V}\pi$ to transitions in δ^π as follows. For any $P \xrightarrow{a} Q$ and L such that $(P, L) \in \Pi \square \mathbb{L}$, take

$$(P, L, Q, a)^\square := \begin{cases} (a, \mathbf{del}(1)) & \text{if } L = [] \wedge (a = x\bullet \vee a = \bar{x}\circ) \wedge \forall \alpha. \alpha \# Q @ \alpha \\ (a, \mathbf{del}()) & \text{otherwise if } L = [] \end{cases}$$

$$\begin{cases} ([1/\alpha]a'^{++}, \psi(\alpha, Q, d')) & \text{if } (L \in \mathbb{L}_0 \vee \forall \alpha \in \mathbb{N}. \alpha \# L @ \alpha) \wedge L = \mathbf{push}(\alpha)(L') \\ & \text{and } (a', d') = (P, L', Q, a)^\square \\ \forall \alpha. (P @ \alpha, L @ \alpha, a @ \alpha, Q @ \alpha)^\square & \text{if } \forall \alpha \in \mathbb{N}. \alpha \# L @ \alpha \\ \forall \alpha. ([1/\alpha]a'^{++}, \psi(\alpha, Q @ \alpha, d')) & \text{if } \forall \alpha \in \mathbb{N}. L @ \alpha = \mathbf{push}(\alpha)(L') \\ & \text{and } (a', d') = (P @ \alpha, L', Q @ \alpha, a @ \alpha)^\square \end{cases}$$

$$\text{where } \psi(x, Q, \mathbf{del}(\vec{m})) := \begin{cases} \mathbf{del}(\vec{m}^{++}) & \text{if } x \# Q \\ \mathbf{del}(1, \vec{m}^{++}) & \text{if } x \# Q \end{cases}$$

Take now (a, d) to be $(P, L, Q, a)^\square$. We can easily show $a[L] = a$.

Now, we observe that $(P, L) \in \Pi \square \mathbb{L}$ implies $(Q, \mathbf{upd}(a, d, L)) \in \Pi \square \mathbb{L}$, since the update d deletes exactly the names in L that do not exist in Q , according to the definition above. For the rest of the proof we proceed by induction on the derivation of $P \xrightarrow{a} Q$. \square

Proof of theorem 26:

Let $\mathcal{A} := \mathcal{A}_{P_0}$, $\mathcal{B} := \mathcal{A}_{Q_0}$ and $\sim_{\mathcal{A}\mathcal{B}}^{\mathcal{B}\mathcal{A}} := \sim_{\mathcal{A}\mathcal{B}} \cup \sim_{\mathcal{B}\mathcal{A}}$.

For the (\Leftarrow) part we proceed as follows. Take

$$\mathcal{R} := \{(P, Q) \in \Pi^2 \mid \exists L, K \in \mathbb{L}. ((P, L)^\square, L) \sim_{\mathcal{A}\mathcal{B}}^{\mathcal{B}\mathcal{A}} ((Q, K)^\square, K)\}$$

It suffices to show that \mathcal{R} is a bisimulation. Note that \mathcal{R} is symmetric. Now, take some $(P, Q) \in \mathcal{R}$ with, say, $((P, L)^\square, L) \sim_{\mathcal{A}\mathcal{B}} ((Q, K)^\square, K)$ and suppose $P \xrightarrow{a} P'$, then,

► If $P \xrightarrow{x\bullet} \hat{P}$, then $(P, L)^\square \circ_{\hat{d}}^{\hat{a}} (\hat{P}, \mathbf{upd}(a, d, L))^\square$, with $(a, d) = (P, L, Q, a)^\square$, $a[L] = x\bullet$ and $\mathbf{add}_a = \mathbf{addfresh}$. Now $((P, L)^\square, L) \sim_{\mathcal{A}\mathcal{B}} ((Q, K)^\square, K)$, so $(Q, K)^\square \circ_{\hat{d}'}^{\hat{a}'}$ Q' , with $a'[K] = a[L] = x\bullet$ and $((\hat{P}, \mathbf{upd}(a, d, L))^\square, \mathbf{upd}(a, d, L)) \sim_{\mathcal{A}\mathcal{B}} (Q', \mathbf{upd}(a', d', K))$. Then, $Q = ((Q, K)^\square, K)^\square \xrightarrow{x\bullet} (Q', \mathbf{upd}(a', d', K))^\square = \hat{Q}$, and $(\hat{Q}, \mathbf{upd}(a', d', K))^\square = Q'$, so $(\hat{P}, \hat{Q}) \in \mathcal{R}$.

Now, if $y \in \mathcal{S}(Q) \setminus \mathcal{S}(P)$ then $y \in \mathcal{S}((Q, K)^\square, K) \setminus \mathcal{S}((P, L)^\square, L)$, so $(Q, K)^\square \circ_{\hat{d}''}^{\hat{a}''}$ Q'' , with $a''[K] = xy$ and $((\hat{P}, \mathbf{d}(\mathbf{addfresh}(L)))^\square, \mathbf{d}(\mathbf{add}(y, L))) \sim_{\mathcal{A}\mathcal{B}} (Q'', \mathbf{upd}(a'', d'', K))$. Now, $Q = ((Q, K)^\square, K)^\square \xrightarrow{xy} (Q'', \mathbf{upd}(a'', d'', K))^\square = Q_y$, and $(Q_y, \mathbf{upd}(a'', d'', K))^\square = Q''$, so it suffices to show that $(\hat{P}, \mathbf{d}(\mathbf{addfresh}(L)))^\square = (\hat{P} @ y, \mathbf{d}(\mathbf{add}(y, L)))^\square$, in order to get $(\hat{P} @ y, Q_y) \in \mathcal{R}$. But, if $\hat{L} = \mathbf{d}(\mathbf{addfresh}(L))$ then $\hat{L} @ y = \mathbf{d}(\mathbf{add}(y, L))$, so it is shown.

► If $P \xrightarrow{xy} P'$, with $y \# Q$, then $(P, L)^\square \circ_{\hat{d}}^{\hat{a}} (P', \mathbf{upd}(a, d, L))^\square$, with $(a, d) = (P, L, P', xy)^\square$ and $a[L] = xy$. Now, $y \# ((Q, K)^\square, K)$, so $(Q, K)^\square \circ_{\hat{d}'}^{\hat{a}'}$ Q' ,

with $\mathbf{a}'[K] = x\bullet$, $\text{add}_{\mathbf{a}'} = \text{addfresh}$ and $((P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus}, \text{upd}(\mathbf{a}, \mathbf{d}, L)) \sim_{\mathcal{AB}} (Q', \mathbf{d}'(\text{add}(y, K)))$.

Now, $Q = ((Q, K)^{\boxplus}, K)^{\boxminus} \xrightarrow{x\bullet} (Q', \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = \hat{Q}$, and $(\hat{Q} @ y, \mathbf{d}'(\text{add}(y, K)))^{\boxplus} = (\hat{Q}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = Q'$, so $(P', \hat{Q} @ y) \in \mathcal{R}$.

► If $P \xrightarrow{a} P'$, with $a \in \text{Act}_0$ and not listed above, work as in the case of $a = x\bullet$.

► If $P \xrightarrow{a} P'$, with $a \notin \text{Act}_0$, then $(P, L)^{\boxplus} \xrightarrow{a} (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus}$, with $(\mathbf{a}, \mathbf{d}) = (P, L, P', a)^{\boxplus}$ and $\mathbf{a}[L] = a$, so $\forall \alpha \in \mathbf{N}. ((P, L)^{\boxplus}, L @ \alpha) \sim_{\mathcal{AB}} ((Q, K)^{\boxplus}, K @ \alpha)$. But now note that $\forall \alpha \in \mathbf{N}. (P @ \alpha, L @ \alpha)^{\boxplus} = (P, L)^{\boxplus} \wedge (Q @ \alpha, K @ \alpha)^{\boxplus} = (Q, K)^{\boxplus}$, thus $\forall \alpha \in \mathbf{N}. (P @ \alpha, Q @ \alpha) \in \mathcal{R}$, as required.

For the (\implies) part we proceed as follows. Take

$$\mathcal{R} := \{((P, L), (Q, K)) \in \mathcal{S}_{\mathcal{A}}^{\boxplus} \mathbb{L} \times \mathcal{S}_{\mathcal{B}}^{\boxplus} \mathbb{L} \mid (P, L)^{\boxplus} \sim (Q, K)^{\boxplus}\}$$

It suffices to show that \mathcal{R} is a bisimulation. Though \mathcal{R} is not symmetric, it suffices to show it is a simulation, since a symmetric argument can be used for bisimulation. So, take some $(P, L) \mathcal{R} (Q, K)$, with $(P, L)^{\boxplus} \sim (Q, K)^{\boxplus}$ and suppose $P \xrightarrow{a} P'$; then, $(P, L)^{\boxplus} \xrightarrow{\mathbf{a}[L]} (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus}$ and,

► If $\mathbf{a}[L] = x\bullet$, then $(Q, K)^{\boxplus} \xrightarrow{x\bullet} \hat{Q} \sim (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus}$ and $\text{add}_{\mathbf{a}} = \text{addfresh}$.

Now, $Q = ((Q, K)^{\boxplus}, K)^{\boxplus} \xrightarrow{a'} (\hat{Q}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = Q'$, with $\mathbf{a}'[K] = x\bullet$ and $((\hat{Q}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = \hat{Q}$, so $(P', \text{upd}(\mathbf{a}, \mathbf{d}, L)) \mathcal{R} (Q', \text{upd}(\mathbf{a}', \mathbf{d}', K))$.

Moreover, if $y \in \mathcal{S}(Q, K) \setminus \mathcal{S}(P, L)$ then $y \in \mathcal{S}((Q, K)^{\boxplus}) \setminus \mathcal{S}((P, L)^{\boxplus})$, so $(Q, K)^{\boxplus} \xrightarrow{xy} Q_y \sim (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus} @ y$. Now, if $\text{upd}(\mathbf{a}, \mathbf{d}, L) = \hat{L}$ then $\mathbf{d}(\text{add}(y, L)) = \hat{L} @ y$, and $(P', \hat{L})^{\boxplus} @ y = \forall \alpha. (P', \langle \alpha \rangle (\hat{L} @ \alpha))^{\boxplus} @ y = \forall \alpha. (\langle \alpha \rangle (P', \hat{L} @ \alpha)^{\boxplus}) @ y = (P', \hat{L} @ y)^{\boxplus}$, as required.

Also, $Q = ((Q, K)^{\boxplus}, K)^{\boxplus} \xrightarrow{a''} (Q_y, \text{upd}(\mathbf{a}'', \mathbf{d}'', K))^{\boxplus} = Q''$, with $\mathbf{a}''[K] = xy$ and $(Q'', \text{upd}(\mathbf{a}'', \mathbf{d}'', K))^{\boxplus} = Q_y$, so $(P', \mathbf{d}(\text{add}(y, L))) \mathcal{R} (Q'', \text{upd}(\mathbf{a}'', \mathbf{d}'', K))$.

► If $\mathbf{a}[L] = xy$ with $y \# Q, K$, then $y \# (Q, K)^{\boxplus}$, so $(Q, K)^{\boxplus} \xrightarrow{x\bullet} \hat{Q}$, with $\hat{Q} @ y \sim (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))^{\boxplus}$. Thus, $Q = ((Q, K)^{\boxplus}, K)^{\boxplus} \xrightarrow{a'} (\hat{Q}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = Q'$, with $\mathbf{a}'[K] = x\bullet$ and $\text{add}_{\mathbf{a}'} = \text{addfresh}$.

Now, $(\hat{Q}, \text{upd}(\mathbf{a}', \mathbf{d}', K))^{\boxplus} = (\hat{Q} @ y, \mathbf{d}'(\text{add}(y, K)))^{\boxplus}$, hence $(Q', \mathbf{d}'(\text{add}(y, K))) \mathcal{R} (P', \text{upd}(\mathbf{a}, \mathbf{d}, L))$.

► If $\mathbf{a}[L] \in \text{Act}_0$ and not listed above, work as in the case of $\mathbf{a}[L] = x\bullet$.

► If $\mathbf{a}[L] \notin \text{Act}_0$, then $\forall \alpha \in \mathbf{N}. (P, L)^{\boxplus} @ \alpha \sim (Q, K)^{\boxplus} @ \alpha$. But $\forall \alpha \in \mathbf{N}. (P, L)^{\boxplus} @ \alpha = (P, L @ \alpha)^{\boxplus} \wedge (Q, K)^{\boxplus} @ \alpha = (Q, K @ \alpha)^{\boxplus}$, thus $\forall \alpha \in \mathbf{N}. (P, L @ \alpha) \mathcal{R} (Q, K @ \alpha)$, as required. \square