A lexically scoped distributed pi-calculus

António Ravara
Ana G. Matos
Vasco T. Vasconcelos
Luís Lopes

DI–FCUL TR–02–4

April 2002

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

A lexically scoped distributed pi-calculus

António Ravara[*] Ana G. Matos[†] Vasco T. Vasconcelos[‡] Luís Lopes[†]

April 2002

Abstract

We define the syntax, the operational semantics, and a type system for *lsd*π, an asynchronous and distributed π-calculus with local communication and process migration. The calculus follows a simple model of distribution for mobile calculi, with a lexical scoping mechanism that provides both for remote communication and for process migration, making explicit migration primitives superfluous.

1 Introduction

Current hardware developments in network technology, namely high-bandwidth, low-latency networks and wireless communication, has opened new prospects for mobile computation, while at the same time introducing new problems that need to be addressed at the software level. The fundamental problem stems from the lack of a formal background on which to assert the correctness of a given system specification. Thus, adequate theoretical modeling of distributed mobile systems is required to produce provably correct software specifications and to reason about distributed computations.

We propose a natural framework to specify such systems, based on the π-calculus [10, 11], with explicit distribution and process migration. Technically, the paper describes an asynchronous and distributed π-calculus, where each free channel belongs to a specific site, fixed throughout the computation, further developing the work in reference [14]. We adhere to the *lexical scoping in a distributed context* of Obliq [5], meaning that, in order to determine where an a certain free channel belongs to, we just have to inspect the code for the process where the channel occurs. The rule is simple: located channels $a@s$ belong to the site where they are (explicitly) located, s; simple channels a are (implicitly) located at the current site, the site where the process they occur at is located.

To motivate the importance of lexical scoping in programming languages in general, consider the following function written in (a variant of) Pascal.

[*]Department of Mathematics, Instituto Superior Técnico. Lisbon, Portugal.
[†]Department of Computer Science, Faculty of Sciences, University of Porto, Portugal.
[‡]Department of Informatics, Faculty of Sciences, University of Lisbon, Portugal.

```
function f (): Integer;
  var x: Integer := 1;
  function g () : Integer;
    begin g := x end;
  begin f := x + g () end;
```

A *programmer* writes the body of function g knowing that x is global to g, and develops function f keeping in mind that x is local to f, and that the x of g and that of f denote the same variable. This is the kind of reasoning that programmers have been doing for decades, both in imperative (Algol, Pascal) and in functional (ML, Haskell) languages. It is intuitive, and accumulated experience has shown the concept to be right. Also, an unoptimizing *compiler* assigns to variable x a memory, together with the remaining information relevant for function f. In order to evaluate the expressions in the body of each function, the code generated must read the value of x: in f it performs a local operation (reading from the data for the current activation), for g it first finds where the f's activation is. The good news is that we do not need to abandon these ideas, when moving into distributed (and code migrating) computing.

Consider a network where we declare a channel x within some site f, ask for the process x?()P to migrate to another site g, and, in parallel launch a receptor x?()Q located at x. Here is the network (in receptive distributed π [2]), and the one obtained after one reduction step.

```
f [new x                          new x@f
   go g. x?()P |                  g [x?()P] ||
   x?()Q]                         f [x?()Q]
```

From the preceding discussion, the pertinent questions are: "where does channel x belong to?", and "where is x to be stored?". Analyzing the new x@f part, it would seem that channel x belongs to site f, but looking at the subnetworks for g and for f we cannot really conclude that. More importantly, we have no clue on where to place the queue for x, for we have receptors for x at both sites g and f. Amadio *et al.* [2] filter out the above networks, by imposing a unique receiver property (as in π_1 [1]), together with other restrictions (locality and receptiveness). Although this is done using a simple type system, it is an extra element that looks to us counter-intuitive and heavy.

We address the above questions by imposing a lexical scope discipline to networks. In *lsd*π, the above left network would reduce to:

```
new x@f
g [x@f?()P'] ||
f [x?()Q]
```

where it is clear (in each of the three lines) that channel x belongs to site f, and that the queue for x should be at f. Process P' (obtained from P by the application of an appropriate substitution) reflects the fact that P has migrated from f to g: free simple channels (implicitly located at f) become attached to their site (channel y becomes y@f); free channels located at the target site g (say z@g) become simple (by dropping the @g part), reflecting their new local status, ready for communication; all the remaining channels remain unchanged. In *lsd*π, programmers may refer to a channel a belonging to a site s by its local name a or by its remote name a@s, reflecting two distinct views of a channel: the local view and the network (global) view. Allowing the two views greatly simplifies the programming task and allows a standard definition for reduction when combined with migration. From an implementation point of view, this explicit notation provides a compiler with precious information to generate code to access channels.

Explicitly located input/output processes (receptors x@f?(y)P and messages x@f!⟨v⟩, absent in every proposal to date) obviate the need for the go primitive, written spawn(s,P) in [1], go s.P in [2], or s::P in [8]. It suffices to attach to such processes the behavior of "migrating toward the site where they belong": a message targeted at x@f must first migrate to f (thus becoming targeted at x) prior to engaging in some communication; a receptor waiting on x@f must first migrate to f (thus waiting now on x) prior to engaging in any communication. In $lsd\pi$, reduction is local, avoiding remote communication between sites in such a ubiquitous operation. This pattern of interaction between clients and servers is an alternative paradigm of growing interest in programming distributed-systems. Clients do not interact remotely with a server. Rather, they move to the site of the server and interact locally until the session ends. Then, they return to their site of origin. Local communication minimizes network traffic and improves scalability. Migration and reduction are intimately related. An input/output process either reduces locally or migrates over the network depending on the prefix. This is highly convenient from an implementation point of view since possible migration operations are clearly marked in the program. Moreover, since the migration units are either input or output processes, their implementation is far easier than it would be for a generic process.

Lexical scope together with compound names introduce subtleties in the definition of free names. Consider the network $r[(\nu\, a)\, a@s!\langle\rangle]$. It is not clear whether channel a belongs to r or to s, as it is not obvious whether $a@s$ should be free or bound. Now consider the network $r[b?(x)x?()(\nu\, a)\, a@s!\langle\rangle]$ interacting with another network $r[b!\langle c@t\rangle]$, yielding $t[c?()(\nu\, a)\, a@s!\langle\rangle]$. We see that a belongs to whatever site x belongs to, possibly neither r nor s (t in this case). We address this problem by imposing *syntactic restrictions* on processes otherwise defined by a context-free grammar (rejecting, among others, the above processes), and we make sure that these restrictions carry through, all the way from alpha-congruence to reduction.

Our values are simple channels a and located channels $a@s$; parameters are simple channels only. This means that we cannot rely on the usual substitution of the π-calculus, where one substitutes channels by channels. To make sure we got the concept right, we start with a general notion of substitution: a total function on names (sites, channels, located channels), defined along the lines of the substitution for the λ-calculus, by Hindley and Seldin [9]. This function is then used to define *name replacement* (used in alpha-congruence), *name instantiation* (the substitution that arises from communication, as in $a!\langle\tilde{v}\rangle \mid a?(\tilde{x})P$), and *name translation* (the substitution that happens during migration, as in $a@s!\langle\tilde{v}\rangle$ or $a@s?(\tilde{x})P$).

For the type system, we take a simplified form of that of Amadio *et al.* [2], which we adapt to deal with the lexical scope of channels. Types for channels are the usual types in the simply typed π-calculus, $Ch(\gamma_1, \dots, \gamma_n)$, describing a channel capable of carrying a series of channels of types $\gamma_1, \ldots, \gamma_n$ [13]. For sites, we only capture the types of the (free) channels at the site: if a_1 to a_n of types γ_1 to γ_n contain the free channels of site s, then we assign to site s the type $\{a_1{:}\gamma_1, \ldots, a_n{:}\gamma_n\}$. The type system is simple and intuitive—a straightforward extension of that for simply typed π-calculus [13]; it assumes types for sites only (types for channels are taken from that of the site the channel belongs to), and enjoys subject-reduction.

Distinctive features of $lsd\pi$ include: separate syntactic categories for processes and networks (in the line of Dπ [8], but unlike $d\pi_1^r$ [2], nomadic-pict [12], the join-calculus [7], and mobile ambients [6]), a syntactically flat structure of the network and local communication (like Dπ, $d\pi_1^r$, but unlike the join-calculus and ambients).

Specifically, the contributions of this work are:

1. a distributed π-calculus that provides for local communication, remote invocation, weak mobility in a lexical scope regime;

2. a rigorous treatment of channels, allowing for the substitution of a channel by a compound channel, ensuring that each channel belongs to a unique, lexically defined, site;

3. a type system revealing the site of each channel, while ensuring subject-reduction.

The rest of the report is organized as follows: the next section presents the syntax of the calculus; the operational semantics is dealt in section 3; section 4 describes types and type assignment; and section 5 compares $lsd\pi$ with related work and points to future developments. Detailed proofs for all the results can be found in the appendix.

2 The calculus

This section presents the (context-free) grammar of the calculus, followed by the syntactic restrictions.

2.1 Syntax

Consider a countable set $\mathcal{C}$ of *simple channels* a, b, c, x, y, z, and a countable set $\mathcal{S}$ of *sites* s, r, t, such that the two sets are disjoint. Compound channels—pairs channel-site, like $a@s$—form *located channels*, designating a channel a at site s, belonging to the set $\mathcal{C}@\mathcal{S} \stackrel{\text{def}}{=} \{a@s \mid a \in \mathcal{C} \wedge s \in \mathcal{S}\}$. Let $u, v, \ldots$ stand for both simple and located channels, henceforth collectively called channels. Take x as a variable ranging over simple channels, $\tilde{x}$ as a sequence of pairwise distinct variables, let $|\tilde{x}|$ denote the length of the sequence $\tilde{x}$, and let $\{\tilde{x}\}$ denote the set of the channels in the sequence $\tilde{x}$; moreover, let $\tilde{v}$ stand for a sequence of channels. Furthermore, let n, m stand for both sites and channels, henceforth collectively called *names*, and belonging to the set $\mathcal{N} \stackrel{\text{def}}{=} \mathcal{C} \cup \mathcal{S} \cup \mathcal{C}@\mathcal{S}$. Finally, let g, h stand for both sites and located channels, henceforth collectively called *global names*.

$$
\begin{array}{rrcllll}
\textit{Simple channels,} & a,b,c,x,y,z & \in & \mathcal{C} & & & \\
\textit{Sites,} & r,s,t & \in & \mathcal{S} & & & \\
\textit{Channels,} & u,v & ::= & a & \mid & a@s & \\
\textit{Globals,} & g,h & ::= & & & a@s & \mid \; s \\
\textit{Names,} & n,m & ::= & a & \mid & a@s & \mid \; s
\end{array}
$$

$$
\begin{array}{rrcl}
\textit{Processes,} & P,Q & ::= & \mathbf{0} \;\mid\; (P \mid Q) \;\mid\; (\nu\, n)\, P \;\mid\; u!\langle\tilde{v}\rangle \;\mid\; u?(\tilde{x})P \\
\textit{Networks,} & N,M & ::= & \mathbf{0} \;\mid\; (N \parallel M) \;\mid\; (\nu\, g)\, N \;\mid\; s[P]
\end{array}
$$

Figure 1: Syntax.

Definition 2.1 (Names, processes and networks). The grammars in figure 1 define the languages of processes and of networks.

Receptors, of the form $u?(\tilde{x})P$, and *messages*, of the form $u!\langle\tilde{v}\rangle$, are the basic processes in the calculus. A receptor is an input-guarded process. A message has a name u for target and carries a sequence of channels $\tilde{v}$ (note that we do not allow to pass sites). The remaining constructors are fairly standard in name-passing process calculi: process $(P \mid Q)$ denotes the *parallel composition* of processes; process $(\nu\, n)\, P$ denotes the *restriction of the scope* of the name n to the process P (often seen as the creation of a new name or site, visible only within P; moreover, if $n \in \mathcal{S}$, no channel explicitly located at that site is visible outside P); and *inaction* $\mathbf{0}$, denotes the terminated process. For the sake of simplicity, we restrict this work to finite processes.

Networks are: processes running at a given site, $s[P]$, where we assume that free simple channels in P are *implicitly located* at s (while located channels are considered to be *explicitly located*); the parallel composition of networks, $(N \parallel M)$, which is simply a merge of networks; the restriction of the scope of a global to a network, $(\nu\, g)\,\mathrm{N}$; and *inaction* $\mathbf{0}$, which denotes the empty network.

As usual in polyadic mobile calculi, we abbreviate $(\nu\, n_1) \cdots (\nu\, n_m)\, P$ to $(\nu\, \tilde{n})\, P$. Let the operator 'ν' extend as far to the right as possible. In $(P \mid Q)$, we omit the parentheses when the meaning is clear.

2.2 Free and bound names

We envisage a "natural" definition for the free names of a process or of a network, according to the classical definitions for the λ-calculus [3, 9], and meeting the intuitions of any π-calculist.

Notation 2.2 (Useful sets). Let $A, B \subseteq \mathcal{N}$.

N	$\mathsf{fn}(N)$	$\mathsf{bn}(N)$
$\mathbf{0}$	$\{\}$	$\{\}$
$(N \parallel M)$	$\mathsf{fn}(N) \cup \mathsf{fn}(M)$	$\mathsf{bn}(N) \cup \mathsf{bn}(M)$
$(\nu\, s)\, N$	$\mathsf{fn}(N) \setminus (\{s\} \cup \mathsf{fn}(N)@s)$	$\mathsf{bn}(N) \cup \{s\} \cup \mathsf{bn}(N)@s$
$(\nu\, a@s)\, N$	$\mathsf{fn}(N) \setminus \{a@s\} \cup \{s\}$	$\mathsf{bn}(N) \cup \{a@s\}$
$s[P]$	$\mathsf{locate}(\mathsf{fn}(P), s) \cup \{s\}$	$\mathsf{locate}(\mathsf{bn}(P), s)$

Figure 2: Free and bound names in networks.

P	$\mathsf{fn}(P)$	$\mathsf{bn}(P)$
$\mathbf{0}$	$\{\}$	$\{\}$
$(P \mid Q)$	$\mathsf{fn}(P) \cup \mathsf{fn}(Q)$	$\mathsf{bn}(P) \cup \mathsf{bn}(Q)$
$(\nu\, s)\, P$	$\mathsf{fn}(P) \setminus (\{s\} \cup \mathsf{fn}(P)@s)$	$\mathsf{bn}(P) \cup \{s\} \cup \mathsf{bn}(P)@s$
$(\nu\, a@s)\, P$	$\mathsf{fn}(P) \setminus \{a@s\} \cup \{s\}$	$\mathsf{bn}(P) \cup \{a@s\}$
$(\nu\, a)\, P$	$\mathsf{fn}(P) \setminus \{a\}$	$\mathsf{bn}(P) \cup \{a\}$
$u!\langle\tilde{v}\rangle$	$\mathsf{names}(u, \tilde{v})$	$\{\}$
$u?(\tilde{x})P$	$\mathsf{fn}(P) \setminus \{\tilde{x}\} \cup \mathsf{names}(u)$	$\mathsf{bn}(P) \cup \{\tilde{x}\}$

Figure 3: Free and bound names in processes.

1. $A@s \overset{\text{def}}{=} \{a@s \mid a \in A \;\vee\; a@s \in A\}$;

2. $a@B \overset{\text{def}}{=} \{a@s \mid s \in B\}$;

3. $\mathsf{names}(n_1 \dots n_m) \overset{\text{def}}{=} \mathsf{names}(n_1) \cup \dots \cup \mathsf{names}(n_m)$,
   where $\mathsf{names}(s) \overset{\text{def}}{=} \{s\}$, $\mathsf{names}(a@s) \overset{\text{def}}{=} \{a@s, s\}$, and $\mathsf{names}(a) \overset{\text{def}}{=} \{a\}$;

4. $\mathsf{locate}(A, s) \overset{\text{def}}{=} A \setminus \mathcal{C} \cup A@s$;

5. $\mathsf{sites}(A) \overset{\text{def}}{=} \{s \mid s \in A \vee a@s \in A\}$.

Moreover, we say that $s \in n$ when $n \in \mathcal{C}@s \cup \{s\}$, and $a \in n$ when $n \in a@\mathcal{S} \cup \{a\}$.

Definition 2.3 (Free and bound names). The rules in Figures 2 and 3 inductively define the sets of *free and bound names* in networks, $\mathsf{fn}(N)$ and $\mathsf{bn}(N)$, and in processes, $\mathsf{fn}(P)$ and $\mathsf{bn}(P)$.

A channel is local to a site if it occurs as a simple channel in that site, or if it occurs explicitly located at that site anywhere in the network. Amongst the binders of the calculus, two cases deserve a special mention: $(\nu\, s)\, N$ makes all free channels local to s invisible outside N; and $(\nu\, a@s)\, N$ creates a new free site s. The free names of a network $s[P]$ are the free names of P where the simple channels are made explicitly located at s, via operator locate.

2.3 Syntactic restrictions

In *lsd*π it is crucial to distinguish local from remote channels. However, the binders may cause undesirable side effects, leading to confusions like those described in the introduction. Therefore, we do not accept all terms resulting from the grammar in Definition 2.1 as processes: we impose syntactic restrictions. To rigorously define them, we use the auxiliary notions of subnetworks and of subprocesses, which result from Definition 2.1.

Remark 2.4 (Subnetworks and subprocesses). One easily defines the set $\mathcal{SN}(N)$ of the *subnetworks* of the network N, and the set $\mathcal{SP}(P)$ of the *subprocesses* of the process P.

We impose two conditions to accept a term as a process:

1. when a located channel has its scope restricted to some process, it cannot be used in that process as a simple channel; and

2. similarly, when a simple channel has its scope restricted to some process, it can not be used in that process as a located channel.

Since these are syntactic conditions, relying on the notions of free and bound names, one can easily state (decidable) properties that capture them. The following definition rigorously states what terms we accept as processes.

Definition 2.5 (Syntactic restrictions).

1. A process P *satisfies the syntactic restrictions*, and we write $P\ \mathsf{ok}$, if for all $Q \in \mathcal{SP}(P)$, we have $Q\ \mathsf{ok}$, and:

   (a) if $Q = (\nu\, a@s)\, R$, then $a \notin \mathsf{fn}(R)$; and

   (b) if $Q = (\nu\, a)\, R$, then $a@\mathcal{S} \cap \mathsf{fn}(R) = \emptyset$; and

   (c) if $Q = u?(\tilde{x})R$, then $\bigcup_{x \in \{\tilde{x}\}} x@\mathcal{S} \cap \mathsf{fn}(Q) = \emptyset$.

2. A network N *satisfies the syntactic restrictions*, and we write $N\ \mathsf{ok}$, if all its subnetworks and all its subprocesses are ok.

In order to understand the need for these syntactic restrictions we have to take into consideration the fundamental ideas of *lsd*π. As in [14], we embody a rule taken from Hennessy and Riely [8] which considers the network $(\nu\, a@s)\, s[P]$ as indistinguishable from the network $s[(\nu\, a)\, P]$ (forthcoming structural congruence rule SN-SCOS$_3$). Also, we fix that $\mathsf{fn}((\nu\, a@s)\, N) = \mathsf{fn}(N) \setminus \{a@s\}$, because we want the binder to capture only the channel a local to site s (and this is a distinctive feature from Dπ, where all free occurrences of a in N are bound). If a process of the form $(\nu\, a)\, P$, where $a@s \in \mathsf{fn}(P)$, was to be accepted, one would have to decide whether $a@s$ is free in such a process.

Therefore, to define the free variables of networks and processes, one should consider three reasonable possibilities:

1. either $\mathsf{fn}((\nu\, a)\, P) = \mathsf{fn}(P) \setminus \{a\}$, where any $a@s$ occurring in P would not be bound by $(\nu\, a)$, and in this way the names a and $a@s$ have no relation to each other;
2. or $\mathsf{fn}((\nu\, a)\, P) = \mathsf{fn}(P) \setminus (\{a\} \cup a@\mathcal{S})$, where all channels a (simple or at some site) would be bound.
3. or $\mathsf{fn}_{\mathrm{s}}((\nu\, a)\, P) = \mathsf{fn}_{\mathrm{s}}(P) \setminus \{a, a@s\}$ where the subscript s would indicate that the considered process appears in site s.

From each of these possibilities, and taking into consideration the above requirements, a contradiction arises:

1. if the first definition is taken, then
   $\mathsf{fn}(s[(\nu\, a)\, a@s!\langle\rangle]) = \{a@s\}$ and $\mathsf{fn}((\nu\, a@s)\, s[a@s!\langle\rangle]) = \emptyset$;
2. if the second definition is taken, then
   $\mathsf{fn}(s[(\nu\, a)\, a@t!\langle\rangle]) = \emptyset$ and $\mathsf{fn}((\nu\, a@s)\, s[a@t!\langle\rangle]) = \{a@t\}$.
3. if the third definition is taken, we will be further presuming that the channel a is to be created also in site s, which might not be the case (the rule applies only to free channels), as the following example shows: $s[b?(x)x?()(\nu\, a)\, a@s!\langle\rangle]$. Depending on the location of the argument received in x, $(\nu\, a)$ may or may not end up in site s. The fundamental idea here is that the location (or creation site) of a restricted channel will be determined only when it is at the top level of a site (e.g., $s[(\nu\, a)\, P]$). This option will offer flexibility in the creation of channels, and is consistent with the rule SN-SCOS$_3$.

In all the three cases, two networks, which are supposed to be structural congruent, have different sets of free variables, thus justifying the need for syntactic restrictions. The introduction of these syntactic restrictions implies more work on the verification of the consistency of the language, for we must prove that networks do not *go wrong*, in the sense that computation does not transform syntactically correct networks into incorrect ones.

3 Operational semantics

This section describes the reduction semantics of *lsd*π, starting from substitution, through alpha-congruence, structural congruence, ending in reduction.

3.1 Substitution

We follow the approach of Hindley and Seldin [9].

Notation 3.1 (Entities). In the sequel, let X denote a network, a process, or a name, and let $\mathcal{X}$ be the set of such entities.

$$\begin{array}{rcll}
\mathbf{0}[\_] & \overset{\text{def}}{=} & \mathbf{0} & \\[4pt]
(N \parallel M)[n/m] & \overset{\text{def}}{=} & N[n/m] \parallel M[n/m] & \\[4pt]
((\nu\, s)\, N)[n/m] & \overset{\text{def}}{=} & (\nu\, s)\, N & \text{if } s \in m \\
((\nu\, s)\, N)[n/m] & \overset{\text{def}}{=} & (\nu\, s)\, N[n/m] & \text{if } s \notin m \text{ and (1)} \\
((\nu\, s)\, N)[n/m] & \overset{\text{def}}{=} & (\nu\, t)\, N[t/s][n/m] & \text{if } s \notin m \text{ and } \neg\text{(1), with } t \text{ fresh} \\[4pt]
((\nu\, a@s)\, N)[n/a@s] & \overset{\text{def}}{=} & (\nu\, a@s)\, N & \\
((\nu\, a@s)\, N)[n/s] & \overset{\text{def}}{=} & (\nu\, a@n)\, N[n/s] & \\
((\nu\, a@s)\, N)[n/m] & \overset{\text{def}}{=} & (\nu\, a@s)\, N[n/m] & \text{if } m \notin \{s, a@s\} \text{ and (2)} \\
((\nu\, a@s)\, N)[n/m] & \overset{\text{def}}{=} & (\nu\, b@s)\, N[b@s/a@s][n/m] & \text{if } m \notin \{s, a@s\} \text{ and } \neg\text{(2), with } b \text{ fresh} \\[4pt]
(s[P])[b@s/a@s] & \overset{\text{def}}{=} & s[P[b/a][b@s/a@s]] & \\
(s[P])[n/s] & \overset{\text{def}}{=} & n[P[n/s]] & \\
(s[P])[n/m] & \overset{\text{def}}{=} & s[P[n/m]] & \text{if } s \notin m
\end{array}$$

$$(1)\ s \notin n \text{ or } m \notin \mathsf{fn}(N)\,; \quad (2)\ n \notin \{a, a@s\} \text{ or } m \notin \mathsf{fn}(N).$$

Figure 4: Substitution on networks.

Definition 3.2 (Substitution). A *substitution on names in an entity* is a total function $\mathcal{X}[\mathcal{N}/\mathcal{N}] \mapsto \mathcal{X}$, inductively defined by the rules in the Figures 4, 5, and 6.

The substitution function gives rise to three different operations on names: *change of bound names*; *communication*; and *migration.* The first will be used to define the alpha-congruence relation and the others to define the reduction relation. To avoid confusion, the application of the substitution function in each of these operations will be referred to as, respectively, *name replacement*, *name instantiation* and *name translation.*

Definition 3.3 (Name replacement/instantiation/translation). Take the finite sequence of substitutions $[n_1/m_1] \cdots [n_k/m_k]$. Then, $\forall i \in \{1, \dots, k\}$, this sequence is a:

1. *name replacement*, if $m_i \in A \Rightarrow n_i \in A$, for $A = \mathcal{C}, \mathcal{S}, \mathcal{C}@s$, for some s.

2. *name instantiation*, if $m_i \in \mathcal{C}$ and $n_i \in \mathcal{C} \cup \mathcal{C}@\mathcal{S}$.

3. *name translation*, if $m_i = a \Rightarrow n_i \in a@\mathcal{S}$ or $m_i \in a@\mathcal{S} \Rightarrow n_i = a$.

Substitution presents two properties, which follow from the reasonable requirement of the names involved in such operations being of the "same nature", as defined above: it commutes with the free names and preserves the syntactic restrictions. In order to establish the first result, it is usefull to extend the notion of substitution.

$$
\begin{array}{rcll}
\mathbf{0}[\_] & \overset{\text{def}}{=} & \text{as in the Networks case} & \\
(P \mid Q)[n/m] & \overset{\text{def}}{=} & \text{as in the Networks case} & \\
((\nu\, s)\, P)[n/m] & \overset{\text{def}}{=} & \text{as in the Networks case} & \\
((\nu\, a@s)\, P)[n/m] & \overset{\text{def}}{=} & \text{as in the Networks case} & \\
((\nu\, a)\, P)[n/a] & \overset{\text{def}}{=} & (\nu\, a)\, P & \\
((\nu\, a)\, P)[n/m] & \overset{\text{def}}{=} & (\nu\, a)\, P[n/m] & \text{if (1) and (2)} \\
((\nu\, a)\, P)[n/m] & \overset{\text{def}}{=} & (\nu\, b)\, P[b/a][n/m] & \text{if (1) and } \neg(2),\ \text{with } b \text{ fresh} \\
(u!\langle v_1 \dots v_n\rangle)[n/m] & \overset{\text{def}}{=} & u[n/m]!\langle v_1[n/m] \dots v_n[n/m]\rangle & \\
(u?(\tilde{x})P)[n/x_i] & \overset{\text{def}}{=} & u[n/x_i]?(\tilde{x})P & \text{if } x_i \in \{\tilde{x}\} \\
(u?(\tilde{x})P)[n/m] & \overset{\text{def}}{=} & u[n/m]?(\tilde{x})P[n/m] & \text{if (3) and (4)} \\
(u?(\tilde{x})P)[n/m] & \overset{\text{def}}{=} & u[n/m]?(x_1..y..x_n)P[y/x_i][n/m] & \text{if (3) and } \neg(4),\ \text{with } y \text{ fresh}
\end{array}
$$

$$(1)\ m \neq a \quad (2)\ a \notin n \text{ or } m \notin \mathsf{fn}(P) \quad (3)\ m \notin \{\tilde{x}\} \quad (4)\ \forall i\colon x_i \notin n \text{ or } m \notin \mathsf{fn}(P)$$

Figure 5: Substitution on processes.

The substitution function $[n/m]$ is expected to transform networks and processes by changing all free occurrences of m by n, but still preserving their structure. One way of observing the changes induced by such substitution is to examine the set of free names before and after the substitution is applied. It is natural to ask whether one can predict those changes just by considering the initial free name set and the pair of names (n, m). The next definition will serve as a tool to prove this result.

Definition 3.4 (Substitution on sets of names). A *substitution of names in a set* $A \subseteq \mathcal{N}$ is a total function $A[\mathcal{N}/\mathcal{N}] \mapsto 2^{\mathcal{N}}$, defined by the rule

$$A[n/m] \overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in A\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in A\})\,.$$

Keep in mind that the Definition 3.4 should be coherent with that of free names. In particular, care should be taken in respect to the two components of located names , for $\mathsf{names}(a@s) \overset{\text{def}}{=} \{a@s, s\}$ (try comparing $\mathsf{fn}(a!\langle\rangle[a@s/a])$ with $\mathsf{fn}(a!\langle\rangle)[a@s/a]$).

Finally, our first result allows us to deal directly with the free names of networks and processes by observing the effects substitution has on them, while abstracting away from the recursive nature of the definition of substitution.

Proposition 3.5 (Substitution commutes with the free names). Let Υ be a finite sequence of substitutions.

1. If X ok and Υ is a name replacement, then $\mathsf{fn}(X\Upsilon) = \mathsf{fn}(X)\Upsilon$.

$$
\begin{array}{rcll@{\qquad}rcll}
(s)[n/s] & \overset{\text{def}}{=} & n & & (a@s)[n/a@s] & \overset{\text{def}}{=} & n & \\
(s)[n/m] & \overset{\text{def}}{=} & s & \text{if } m \neq s & (a@s)[n/s] & \overset{\text{def}}{=} & a@n & \\
(a)[n/a] & \overset{\text{def}}{=} & n & & (a@s)[n/m] & \overset{\text{def}}{=} & a@s & \text{if } m \notin \{s, a@s\} \\
(a)[n/m] & \overset{\text{def}}{=} & a & \text{if } m \neq a & & & &
\end{array}
$$

Figure 6: Substitution on names.

2. If P ok and Υ is a name instantiation, then $\mathsf{fn}(P\Upsilon) = \mathsf{fn}(P)\Upsilon$.
3. If P ok and Υ is a name translation, then $\mathsf{fn}(P\Upsilon) \subseteq \mathsf{fn}(P)\Upsilon$.

Proof. In each case, the proof consists in a structural induction over the entities, together with a mathematical induction over the length of the sequence of substitutions. The following results are useful auxiliary lemmas.

1. For $\mathcal{A}_1, \ldots, \mathcal{A}_k \subseteq \mathcal{N}$, $(\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_k)[n/m] = \mathcal{A}_1[n/m] \cup \ldots \cup \mathcal{A}_k[n/m]$.
2. For X ok, if $a@t \in \mathsf{fn}(X)$ then $t \in \mathsf{fn}(X)$. Consequently, $\mathsf{sites}(\mathsf{fn}(X)) \subseteq \mathsf{fn}(X)$.
3. For $\mathcal{A} \subseteq \mathcal{N}$, if $[n/m]$ is a name replacement then $\mathcal{A}[n/m] = \{m_1[n/m] \mid m_1 \in \mathcal{A}\}$.

A detailed proof can be found in page 27. □

Due to the context in which the substitution operations occur, the below results apply to both networks and processes for the name replacement, and to processes only for the name instantiation and translation. The reason for the weaker proposition in the translation case is a consequence of the definition of the names in a compound name, and the fact that the translation of a compound name might be a simple name: $\mathsf{fn}(a@t!\langle\rangle[a/a@t]) \overset{\text{def}}{=} \mathsf{fn}(a!\langle\rangle) \overset{\text{def}}{=} \{a\}$, but $\mathsf{fn}(a@t!\langle\rangle)[a/a@t] \overset{\text{def}}{=} \{a@t, t\}[a/a@t] \overset{\text{def}}{=} \{a, t\}$.

Proposition 3.6 (Substitution preserves the syntactic restrictions). Let Υ be a finite sequence of substitutions.

1. If X ok and Υ is a name replacement, then $X\Upsilon$ ok.
2. If P ok and Υ is a name instantiation, then $P\Upsilon$ ok.
3. If P ok and Υ is a name translation, then $P\Upsilon$ ok.

Proof. Again, in each case the proof consists in a structural induction over the entities, interleaved with mathematical induction over the length of the sequence of substitutions. Proposition 3.5 is invoked in the cases where syntactic restriction requirements must be verified, i.e. $(\nu\, a@s)\, \mathrm{X}$, $(\nu\, a)\, \mathrm{X}$, and $u?(\tilde{x})P$. A detailed proof is in page 43. □

3.2 Alpha congruence

The above definition of substitution allows a simple definition of *alpha congruence*, using an auxiliary operation, called *change of bound name* (briefly, *c.o.b.n.*).

Definition 3.7 (Change of bound name).

1. Networks: N is obtained from M by a *c.o.b.n.*, if N is obtained from M by replacing some subnetwork M' of M by N' such that, either:

   (a) $M' = ((\nu\, s)\, M'')$ and $N' = (\nu\, t)\, M''[t/s]$ and $t \notin \mathsf{fn}(M'')$, or

   (b) $M' = ((\nu\, a@s)\, M')$ and $N' = (\nu\, c@s)\, M''[c@s/a@s]$ and $c@s \notin \mathsf{fn}(M'')$.

2. Processes: Q is obtained from P by a *c.o.b.n.*, if Q is obtained from P by replacing some subprocess P' of P by Q' such that, either:

   (a) $P' = (\nu\, s)\, P''$ and $Q' = (\nu\, t)\, P''[t/s]$ and $t \notin \mathsf{fn}(P'')$, or

   (b) $P' = (\nu\, a@s)\, P''$ and $Q' = (\nu\, c@s)\, P''[c@s/a@s]$ and $c, c@s \notin \mathsf{fn}(P'')$, or

   (c) $P' = (\nu\, a)\, P''$ and $Q' = (\nu\, c)\, P''[c/a]$ and $c \notin \mathsf{fn}(P'')$ and $c@\mathcal{S} \cap \mathsf{fn}(P'') = \emptyset$, or

   (d) $P' = u?(x_1 \ldots y \ldots x_n)P''$ and $Q' = u?(x_1 \ldots z \ldots x_n)P''[z/y]$ and $z \notin \mathsf{fn}(P'')$ and $z@\mathcal{S} \cap \mathsf{fn}(P'') = \emptyset$ and $z \notin \{x_1 \ldots x_n\}$.

Using this auxiliary definition, it is now easy to define alpha-congruence, as follows.

Definition 3.8 (Alpha congruence).

1. Networks: $N \equiv_\alpha M$ if N is obtained from M by a series of *c.o.b.n.*

2. Processes: $P \equiv_\alpha Q$ if P is obtained from Q by a series of *c.o.b.n.*

To understand the mechanism by which this definition prevents ill-matched networks and processes to be related by alpha congruence, first consider the two cases which occur classically in the λ-calculus, and are prevented here in the same way. In fact, the name capture which could arise by changing $(\nu\, t)$ to $(\nu\, s)$ in the process $(\nu\, t)\, (\nu\, a@s)\, a@t!\langle\rangle$ is prevented by the definition of substitution invoked by rule 3.7.2a. Notice that the definition of substitution changes the innermost binder name to a fresh name. On the other hand, the change of $(\nu\, b@s)$ to $(\nu\, a@s)$ in network $(\nu\, b@s)\, s[a!\langle\rangle]$ is prevented by the side condition of the applicable rule 3.7.1b.

If not treated carefully, alpha congruence could lead syntactically correct processes into incorrect ones, for the same reasons as the above two classes of problems. These are prevented using the same mechanisms, but let us concentrate on the alpha congruence side conditions:

1. To prevent process $(\nu\, a@s)\, c!\langle\rangle$ to be alpha congruent to $(\nu\, c@s)\, c!\langle\rangle$ (thus conflicting with syntactic restriction 2.5.1a), in order to be able to apply rule 3.7.2b, the condition $c \notin \mathsf{fn}(c!\langle\rangle)$ must be verified.

[SN-ALPHA]	$N \equiv M$	if $N \equiv_\alpha M$
[SN-ASSO]	$((N \parallel M) \parallel M') \equiv (M \parallel (N \parallel M'))$	
[SN-COMM]	$(M \parallel N) \equiv (N \parallel M)$	
[SN-NEUT]	$(N \parallel \mathbf{0}) \equiv N$	
[SN-SCOP]	$((\nu\, g)\, N) \parallel M \equiv (\nu\, g)\, (N \parallel M)$	if $g \notin \mathsf{fn}(M)$
[SN-RESO]	$(\nu\, g)\, (\nu\, h)\, N \equiv (\nu\, h)\, (\nu\, g)\, N$	if $g \notin \mathsf{names}(h)$ and $h \notin \mathsf{names}(g)$
[SN-RESZ]	$(\nu\, g)\, \mathbf{0} \equiv \mathbf{0}$	
[SN-SCOS$_1$]	$(\nu\, r)\, s[P] \equiv s[(\nu\, r)\, P]$	if $r \neq s$
[SN-SCOS$_2$]	$(\nu\, a@r)\, s[P] \equiv s[(\nu\, a@r)\, P]$	if $a \notin \mathsf{fn}(P)$
[SN-SCOS$_3$]	$(\nu\, a@s)\, s[P] \equiv s[(\nu\, a)\, P]$	if $a@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$
[SN-ROUT]	$(s[P] \parallel s[Q]) \equiv s[P \mid Q]$	
[SN-INAC]	$s[\mathbf{0}] \equiv \mathbf{0}$	
[SN-MIGO]	$s[a@s!\langle\tilde{v}\rangle] \equiv s[a!\langle\tilde{v}\rangle]$	
[SN-MIGI]	$s[a@s?(\tilde{x})P] \equiv s[a?(\tilde{x})P]$	

Figure 7: Structural congruence on networks.

2. Similarly, to prevent process $(\nu\, a)\, c@s!\langle\rangle$ to be alpha congruent to $(\nu\, c)\, c@s!\langle\rangle$ (violating syntactic restriction 2.5.1b), in order to be able to apply rule 3.7.2c, the condition $c@\mathcal{S} \cap \mathsf{fn}(c@s!\langle\rangle) = \emptyset$ must be verified.

The following result ensures that the alpha-congruence relation is free of such problems.

Proposition 3.9 (Alpha congruence preserves the free names and the syntactic restrictions). Let X and Y be both either networks or processes.

1. If X ok and $X \equiv_\alpha Y$, then $\mathsf{fn}(X) = \mathsf{fn}(Y)$.
2. If X ok and $X \equiv_\alpha Y$, then Y ok.

Proof. Both proofs use the name replacement version of Proposition 3.5 for a verification on each case of *c.o.b.n.*. Note that all the substitutions involved in the definition of *c.o.b.n.* are name replacements. Furthermore, the second proof uses the name replacement version of Proposition 3.6. Check the details in page 55. □

3.3 Structural congruence

As usual in process calculi, we define the operational semantics of *lsd*π following a “chemical style” [4], i.e., via two binary relations on entities: a static one—*structural congruence*—and a dynamic one—*reduction*.

[SP-ALPHA]	$P \equiv Q$	if $P \equiv_\alpha Q$
[SP-ASSO]	$((P \mid Q) \mid R) \equiv (P \mid (Q \mid R))$	
[SP-COMM]	$(P \mid Q) \equiv (Q \mid P)$	
[SP-NEUT]	$(P \mid \mathbf{0}) \equiv P$	
[SP-SCOP$_1$]	$((\nu\, s)\, P) \mid Q \equiv (\nu\, s)\,(P \mid Q)$	if $s \notin \mathsf{fn}(Q)$
[SP-SCOP$_2$]	$((\nu\, a@s)\, P) \mid Q \equiv (\nu\, a@s)\,(P \mid Q)$	if $a, a@s \notin \mathsf{fn}(Q)$
[SP-SCOP$_3$]	$((\nu\, a)\, P) \mid Q \equiv (\nu\, a)\,(P \mid Q)$	if $(\{a\} \cup a@\mathcal{S}) \cap \mathsf{fn}(Q) = \emptyset$
[SP-RESO$_1$]	$(\nu\, g)\,(\nu\, h)\, P \equiv (\nu\, h)\,(\nu\, g)\, P$	if $g \notin \mathsf{names}(h)$ and $h \notin \mathsf{names}(g)$
[SP-RESO$_2$]	$(\nu\, a)\,(\nu\, s)\, P \equiv (\nu\, s)\,(\nu\, a)\, P$	if $a@s \notin \mathsf{fn}(P)$
[SP-RESO$_3$]	$(\nu\, a)\,(\nu\, u)\, P \equiv (\nu\, u)\,(\nu\, a)\, P$	if $u \notin a@\mathcal{S}$
[SP-RESZ]	$(\nu\, n)\, \mathbf{0} \equiv \mathbf{0}$	

Figure 8: Structural congruence on processes.

Definition 3.10 (Structural congruence). The *structural congruence relation* is the least congruence relation containing the rules in the Figures 7 and 8.

The rules in Figure 7 are inspired on those proposed by Hennessy and Riely for Dπ [8], while those in Figure 8 are adapted from the standard rules of the π-calculus [10, 11]. Some of these rules deserve a special mention:

1. Rule SN-ROUT describes the way by which processes within a site may be split or aggregated.

2. Rule SN-INAC garbage collects inactive sites.

3. The rules SCOP, RESO, RESZ (both SN and SP), and SCOS define the scope a binder may take: the scope can expand and contract in such a way that no name is captured or released, and no syntactic conflict arises. Furthermore, rules SCOS extend this principle to suit the definition of free names.

4. The rules SN-MIGO and SN-MIGI embody the notion that simple channels always belong to the site where the process is running. This clarification is needed only at communication time, when the channel is actually used for reduction (see rule RP-COMM in figure 9).

To clarify the side conditions of the rule RESO, we present examples of pathological cases that are excluded.

1. The network or process $(\nu\, a@s)\,(\nu\, s)\, X$ should not be congruent to $(\nu\, s)\,(\nu\, a@s)\, X$, for the restricted located channel $(\nu\, a@s)$ is either being captured or released during the commutation of the ν.

2. The process $(\nu\, a)\,(\nu\, s)\, a@s!\langle\rangle$ obviously should not be congruent to $(\nu\, s)\,(\nu\, a)\, a@s!\langle\rangle$, since the latter process is not ok.

3. The process $(\nu\, a)\,(\nu\, a@s)\, a@s!\langle\rangle$ should not be congruent to $(\nu\, a@s)\,(\nu\, a)\, a@s!\langle\rangle$, and similarly, $(\nu\, a@s)\,(\nu\, a)\, a!\langle\rangle$ should not be congruent to $(\nu\, a)\,(\nu\, a@s)\, a!\langle\rangle$, since in both cases the resulting processes are not ok.

Proposition 3.11 (Structural congruence preserves the free names and the syntactic restrictions). Let X and Y both be either networks or processes.

1. If X ok and $X \equiv Y$, then $\mathsf{fn}(X) = \mathsf{fn}(Y)$.

2. If X ok and $X \equiv Y$, then Y ok.

Proof. Both proofs perform a verification on each rule of structural congruence. In the process of proving this proposition, the side conditions of the mentioned rules may be understood. The cases of the ALPHA rules follow from Proposition 3.9. Check the details in page 60. □

3.4 Simultaneous substitutions and name translation

Since the calculus is polyadic, it is necessary to substitute several names at a time.

Definition 3.12 (Simultaneous substitutions). Consider two subsets $\{u_1, \dots, u_n\}$ and $\{v_1, \dots, v_n\}$ of $\mathcal{C}$, such that $u_1, \dots, u_n$ are pairwise distinct. Then $\{v_1/u_1, \dots, v_n/u_n\}$ is a *set of simultaneous substitutions.* The result of applying this set of substitutions to P is

$$P\{v_1/u_1, \dots, v_n/u_n\} \overset{\text{def}}{=} \begin{cases} P[v_1/u_1]\cdots[v_n/u_n] & \text{if } \{\tilde{u}\} \cap \{\tilde{v}\} = \emptyset\,, \\ P[w_1/u_1]\cdots[w_n/u_n][v_1/w_1]\cdots[v_n/w_n] & \text{otherwise,} \end{cases}$$

where $w_1, \dots, w_n$ are fresh.

Notation 3.13 (Set of simultaneous substitutions). Let $\{\tilde{v}/\tilde{x}\} \overset{\text{abv}}{=} \{v_1/x_1, \dots, v_n/x_n\}$ denote a set of simultaneous substitutions.

One can easily show that the order in which the substitutions in these sets are written is irrelevant to the result.

Proposition 3.14 (Commutativity in a set of simultaneous substitutions). $P\{v_1/u_1, \dots, v_n/u_n\} = P\{v'_1/u'_1, \dots v'_n/u'_n\}$ modulo *c.o.b.n.*, where $\{v'_1/u'_1, \dots v'_n/u'_n\}$ is a permutation of $\{v_1/u_1, \dots, v_n/u_n\}$.

Proof. By a simple induction on the cardinality of the set. □

Both the communication and the migration primitives that we present below, are defined using the above notion of set of simultaneous substitutions. Notice that, in order to avoid name capture and release, as well as the emergence of syntactic errors, renaming of bound names may occur. This may be seen as an "automatic alpha-conversion" that enables substitution to be a total function.

To keep channels "local by default", when migrating process P from site r to site s, the free channels of P must be renamed accordingly: thus simple channels become explicitly located at r; channels located at s become simple (dropping the $@s$ part); all other channels remain unchanged.

Definition 3.15 (Name translation). Let $A \subseteq \mathcal{N}$, $A \cap \mathcal{C} = \{a_1, \ldots, a_n\}$ and $A \cap \mathcal{C}@s = \{b_1@s, \ldots, b_m@s\}$. Then,

$$\sigma(A, r, s) \stackrel{\text{def}}{=} \{a_1@r/a_1, \ldots, a_n@r/a_n, b_1/b_1@s, \ldots, b_m/b_m@s\}\,.$$

Notation 3.16 (Application of name translation to a process). Let $P\sigma_{rs}$ abbreviate the result of applying name translation $\sigma(\mathsf{fn}(P), r, s)$ to process P.

3.5 Reduction

We are finally in a position to define the reduction relation. Reduction contexts simplify the presentation of the reduction relation.

Definition 3.17 (Reduction contexts).

$$\begin{array}{lclcccc} E & ::= & [] & \mid & (E \mid P) & \mid & (\nu\, n)\, E \\ F & ::= & [] & \mid & (F \parallel N) & \mid & (\nu\, g)\, F \end{array}$$

Definition 3.18 (Reduction). The rules in the Figure 9 inductively define the *reduction relation* on processes and networks.

Axiom RP-COMM is standard in the π-calculus [10, 11], the axioms RN-MIGO and RN-MIGI were proposed by Vasconcelos *et al.* for DiTyCO [14], and the remaining rules in the Figure 9 were proposed by Amadio *et al.* for Dπ [2].

So far we have defined two main relations (structural congruence and reduction) that allow us to rewrite programs and simulate their computation. The two main results of this section guarantee that, during reduction, on the one hand the set of free names does not increase, and on the other hand, syntactic correctness is preserved.

Proposition 3.19 (Reduction preserves the free names and the syntactic restrictions). Let X and Y both be either networks or processes.

1. If X ok and $X \to Y$, then $\mathsf{fn}(X) \supseteq \mathsf{fn}(Y)$.

2. If X ok and $X \to Y$, then Y ok.

$$\begin{array}{rll}
\text{[RP-COMM]} & a?(\tilde{x})P \mid a!\langle\tilde{v}\rangle \to P\{\tilde{v}/\tilde{x}\} & \\
\text{[RN-MIGO]} & r[a@s!\langle\tilde{v}\rangle] \to s[(a@s!\langle\tilde{v}\rangle)\sigma_{rs}] & r \neq s \\
\text{[RN-MIGI]} & r[a@s?(\tilde{x})P] \to s[(a@s?(\tilde{x})P)\sigma_{rs}] & r \neq s \\
\text{[RP-CONT]} & \dfrac{P \to Q}{E[P] \to E[Q]} & \\
\text{[RP-STR]} & \dfrac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q} & \\
\text{[RN-SITE]} & \dfrac{P \to Q}{s[P] \to s[Q]} & \\
\text{[RN-CONT]} & \dfrac{N \to M}{F[N] \to F[M]} & \\
\text{[RN-STR]} & \dfrac{N \equiv N' \quad N' \to M' \quad M' \equiv M}{N \to M} &
\end{array}$$

Figure 9: Reduction rules.

Proof. Both proofs consist in an induction on the derivation of the reduction step. The delicate cases are the axioms of reduction RP-COMM, RN-MIGO and RN-MIGI. An auxiliary lemma is useful: If $\mathcal{A} \subseteq \mathcal{N}$ and Υ is a finite sequence of substitutions, then

$$\mathcal{A}\Upsilon = \{m\Upsilon \mid m \in \mathcal{A}\} \cup \mathsf{sites}(\{m\Upsilon \mid m \in \mathcal{A}\})\,.$$

Use Proposition 3.5 and the lemma to prove the first clause, and Proposition 3.6 to prove the second. The induction steps concerning rules STR use Proposition 3.11. In the cases of CONT a second induction on the structure of the contexts should be used. Check the details in page 72. $\square$

3.6 Examples

We proceed by presenting some examples of the use of the language. We omit the application of some rules, like SN-COMM and SP-COMM, and underline redexes.

1. As an academic example, consider a site s running two processes, P and Q, in parallel. Process $P = (\nu\, a)\,(b@t?(x)x?()\mathbf{0} \mid a?()\mathbf{0})$ is expecting a message at a channel b located at site t. Furthermore, site t is running a process ready to send a message on a local

channel with the same channel b.

$$s[(\nu\, a)\,(b@t?(x)x?()\mathbf{0} \mid a?()\mathbf{0}) \mid Q] \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

The sphere of action of P is restricted to the scope of the enfolding $(\nu\, a)$, which is internal to site s, but using SN-SCOS, we may extrude it to the network level.

$$\equiv \quad (\nu\, a@s)\,(s[b@t?(x)(x?()\mathbf{0} \mid a?()\mathbf{0}) \mid Q]) \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

In order for P to migrate to site t, where it can perform the communication, it must be isolated, so, using SN-ROUT, we separate the site s into two parts.

$$\equiv \quad (\nu\, a@s)\,(\underline{s[b@t?(x)(x?()\mathbf{0} \mid a?()\mathbf{0})]} \parallel s[Q]) \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

Process P is now ready to migrate (using RN-MIGI).

$$\rightarrow \quad (\nu\, a@s)\,(t[b?(x)(x?()\mathbf{0} \mid a@s?()\mathbf{0})] \parallel s[Q]) \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

Within the current scope of $(\nu\, a@s)$, no communication can occur at channel b, even though the two process are located at site t. We wish to extrude the scope of $(\nu\, a@s)$ even further, to encompass all the fragments of site t. Since $a \in \mathsf{fn}(b!\langle a\rangle)$, this will only be possible if we rename the bound $a@s$ to a fresh name, say $c@s$, using SN-ALPHA. In this way, there will be no confusion between the channel currently named "a", and other uses of the same name.

$$\equiv_\alpha \quad (\nu\, c@s)\,(t[b?(x)(x?()\mathbf{0} \mid c@s?()\mathbf{0})] \parallel s[Q][c@s/a@s]) \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

Now we are free to expand the scope of $(\nu\, c@s)$ (using SN-SCOP),

$$\equiv \quad (\nu\, c@s)\, t[b?(x)(x?()\mathbf{0} \mid c@s?()\mathbf{0})] \parallel s[Q][c@s/a@s] \parallel t[b!\langle a\rangle \mid a@s!\langle\rangle \mid a!\langle\rangle]$$

and with SN-ROUT, we merge the two fragments of site t.

$$\equiv \quad (\nu\, c@s)\, t[\underline{b?(x)(x?()\mathbf{0} \mid c@s?()\mathbf{0})} \mid \underline{b!\langle a\rangle} \mid a@s!\langle\rangle \mid a!\langle\rangle] \parallel s[Q][c@s/a@s]$$

Communication may now proceed.

$$\rightarrow \quad (\nu\, c@s)\, t[\underline{a?()\mathbf{0}} \mid c@s?()\mathbf{0} \mid a@s!\langle\rangle \mid \underline{a!\langle\rangle}] \parallel s[Q][c@s/a@s]$$

Now observe which processes are entitled to communicate. Only one process reduction is possible.

$$\rightarrow \quad (\nu\, c@s)\, t[c@s?()\mathbf{0} \mid a@s!\langle\rangle] \parallel s[Q][c@s/a@s]$$

2. A remote procedure call as in reference [14]. The client at site s uses the channel p to invoke a procedure Q at site r with a local argument v (assume that a does not occur free in Q), waits for the reply and continues with P. The reply carries a local name u, which is sent by procedure Q at the end of its computation.

$$\begin{array}{rcl}
s[(\nu\, a)\, (p{@}r!\langle v\ a\rangle \mid a?(y)P)] \parallel r[p?(x\ r)Q] & \equiv & [(1)] \\
(\nu\, a{@}s)\, \underline{s[p{@}r!\langle v\ a\rangle]} \parallel s[a?(y)P] \parallel r[p?(x\ r)Q] & \rightarrow & [\text{RN-MIGO}] \\
(\nu\, a{@}s)\, r[p!\langle v{@}s\ a{@}s\rangle] \parallel s[a?(y)P] \parallel r[p?(x\ r)Q] & \equiv & [\text{SN-ROUT}] \\
(\nu\, a{@}s)\, s[a?(y)P] \parallel r[\underline{p?(x\ r)Q} \mid \underline{p!\langle v{@}s\ a{@}s\rangle}] & \rightarrow & [\text{RP-COMM}] \\
(\nu\, a{@}s)\, s[a?(y)P] \parallel r[\underline{Q\{v{@}s\ a{@}s/x\ r\}}] & \ldots & [Q \text{ reduces}] \\
(\nu\, a{@}s)\, s[a?(y)P] \parallel r[\underline{a{@}s!\langle u\rangle}] & \rightarrow & [\text{RN-MIGO}] \\
(\nu\, a{@}s)\, s[a?(y)P] \parallel s[a!\langle u{@}r\rangle] & \equiv & [\text{SN-ROUT}] \\
(\nu\, a{@}s)\, s[\underline{a?(y)P} \mid \underline{a!\langle u{@}r\rangle}] & \rightarrow & [\text{RP-COMM}] \\
(\nu\, a{@}s)\, s[P\{u{@}r/y\}] & \equiv & [\text{SN-SCOS}] \\
s[(\nu\, a)\, P\{u{@}r/y\}] & &
\end{array}$$

(1) SN-SCOS, SN-ROUT, and SN-SCOP.

3. A primitive for the migration of arbitrary processes, under the lexical scope regime. To send a process P from site r to site s, one creates a remote channel $a{@}s$ (not free in P), prefix P with a receptor on $a{@}s$ and put in parallel an message to $a{@}s$.

$$\begin{array}{rcl}
\mathsf{go}\ s.P \overset{\text{abv}}{=} r[(\nu\, a{@}s)\, a{@}s?()P \mid a{@}s!\langle\rangle] & \equiv & [\text{SN-SCOP}, \text{SN-ROUT}] \\
(\nu\, a{@}s)\, \underline{r[a{@}s?()P]} \parallel \underline{r[a{@}s!\langle\rangle]} & \rightarrow^2 & [\text{SN-MIGO}, \text{SN-MIGI}] \\
(\nu\, a{@}s)\, s[a?()P\sigma_{rs}] \parallel s[a!\langle\rangle] & \equiv & [\text{SN-ROUT}, \text{SN-SCOP}] \\
s[(\nu\, a)\, \underline{a?()P\sigma_{rs}} \mid \underline{a!\langle\rangle}] & \rightarrow & [\text{SN-COMM}] \\
s[P\sigma_{rs}] & &
\end{array}$$

Notice that the process that ends up in site s is not P but P with names translated by σ. Contrast with $\mathsf{go}\ s.P \rightarrow s[P]$ in Amadio *et al.* [2].

4. The creation of "subsites". One may create a (logical) subsite and restrict its access to authorized processes. Since the name of this subsite is private to the master site, external processes must be given its identity to migrate there. In the example below, site s creates a subsite r and a channel $c{@}r$, communicates this name to site t that uses it to download process Q (where $x \notin \mathsf{fn}(Q)$) from the t to r.

$$\begin{array}{rl}
s[(\nu\, r)\, (\nu\, c{@}r)\, \underline{a{@}t!\langle c{@}r\rangle} \mid P] \parallel t[a?(x)x?()Q \mid R] & \rightarrow \\
(\nu\, r)\, (\nu\, c{@}r)\, s[P] \parallel t[\underline{a!\langle c{@}r\rangle} \mid \underline{a?(x)x?()Q} \mid R] & \rightarrow \\
(\nu\, r)\, (\nu\, c{@}r)\, s[P] \parallel t[\underline{c{@}r?()Q} \mid R] & \rightarrow \\
(\nu\, r)\, (\nu\, c{@}r)\, (r[c?()Q\sigma_{tr}] \parallel s[P]) \parallel t[R] &
\end{array}$$

5. The creation of channels "anywhere" in the network. Consider a server with address a at site s that provides some application, which requires some resources (say a private name b). Both local and remote clients may download it, since process

$$s[a?(x)x?()((\nu\, b)\, P) \mid\; (a!\langle c\rangle \mid c!\langle\rangle) \;\mid\; (a!\langle c@r\rangle \mid c@r!\langle\rangle)]$$

reduces either to $s[(\nu\, b)\, P]$ or to $r[(\nu\, b)\, P]$ (consider that x does not occur free in P). Therefore, the site where $(\nu\, b)\, P$ will end up in is determined only at run-time.

This example illustrates an advantage of maintaining both simple and located forms of channels. If we were not able to specify the creation of simple channels (like b), then, since we don't allow the passing of site names, all the locations of restricted channels would be determined statically. At first it might seem as an entanglement, but we believe that, in result of this decision, we can do without the passing of sites.

6. The creation of remote channels might be an undesirable operation. A possibility is to disallow the $(\nu\, a@s)\, P$ constructor. Then we would only be able to migrate processes onto sites on which we know a friend that lends new channels (this is the approach of reference [14]). Below, site r knows $\mathsf{friend}@s$, asks for a new channel c (at s), and prefixes the process to migrate P at the new channel $c@s$.

$$\begin{aligned}(\nu\, \mathsf{friend}@s)\, r[(\nu\, a)\, a?(x)x?()P \mid \underline{\mathsf{friend}@s!\langle a\rangle}] \parallel s[\mathsf{friend}?(x)((\nu\, c)\, x!\langle c\rangle \mid c!\langle\rangle)] \to\\ (\nu\, \mathsf{friend}@s, a@r)\, r[a?(x)x?()P] \parallel s[\underline{\mathsf{friend}!\langle a@r\rangle} \mid \underline{\mathsf{friend}?(x)((\nu\, c)\, x!\langle c\rangle \mid c!\langle\rangle)}] \to\\ (\nu\, a@r)\, r[a?(x)x?()P] \parallel s[(\nu\, c)\, \underline{a@r!\langle c\rangle} \mid c!\langle\rangle] \to\\ (\nu\, c@s)\, r[(\nu\, a)\, \underline{a?(x)x?()P} \mid \underline{a!\langle c@s\rangle}] \parallel s[c!\langle\rangle)] \to\\ (\nu\, c@s)\, r[c@s?()P] \parallel s[c!\langle\rangle)]\end{aligned}$$

At this point $c@s?()P$ may migrate to the friend's site s, where a trigger $c!\langle\rangle$ awaits.

$$\begin{aligned}(\nu\, c@s)\, r[\underline{c@s?()P}] \parallel s[c!\langle\rangle)] \to\\ s[(\nu\, c)\, \underline{c?()P\sigma_{rs}} \mid \underline{c!\langle\rangle})] \to\\ s[P\sigma_{rs}]\end{aligned}$$

The six reduction steps may be classified as three remote operations, each composed of migration followed by local reduction.

Some of the features described above should be carefully used. Type systems may be used to control and discipline the behavior of programs. The following section presents a first system, a very basic one still not addressing security issues, but already ensuring the absence of run-time errors.

4 The type system

This section presents the syntax of types, and a type checking system for *lsd*π. The system is a straightforward extension of that for the simply typed π-calculus [13], but also borrowing ideas from that of Amadio *et al.*'s $d\pi_1^r$ [2]; it assumes types for sites only (types for channels are taken from that of the site the channel belongs to), and enjoys subject-reduction.

An essential ingredient of $d\pi_1^r$, a receptive and asynchronous version of Dπ, is the type system, a simplified version of that of Hennessy and Riely [8], although it types less processes (but a simple extension of the notion of type would probably lead to equivalent systems). Types of *lsd*π are a subset of those of $d\pi_1^r$: simply remove the located type $\gamma^@$, as a located channel may substitute a simple channel. The typing rules were adapted to take into consideration the lexical scope of channels.

Definition 4.1 (Types). Let $n \geq 0$ and let $a_1, \ldots, a_n$ and also $s_1, \ldots, s_n$ be pairwise distinct.

$$\begin{array}{rcl}
\text{Channel types,}\ \gamma & ::= & Ch(\gamma_1, \ldots, \gamma_n) \\
\text{Site types,}\ \varphi & ::= & \{a_1{:}\gamma_1, \ldots, a_n{:}\gamma_n\} \\
\text{Typings,}\ \Gamma & ::= & \{s_1{:}\varphi_1, \ldots, s_n{:}\varphi_n\}
\end{array}$$

We have channel types (those of simple and located channels), and site types (those of sites). A site type is a partial function from channels into channel types. A typing is a partial function from sites into site types.

Notation 4.2. 1. Let F, G be maps. The domain of F is written $\mathrm{dom}(F)$; the map obtained from F by removing x from its domain is denoted by $F \setminus x$. The disjoint union of F and G is denoted by $F \uplus G$.

2. Consider the union of typing assumptions $\Gamma+\Delta$ defined pointwise as, for all $s\in\mathrm{dom}(\Gamma)$ and for all $a \in \Gamma(s) \cap \Delta(s)$:

$$(\Gamma + \Delta)(s) \stackrel{\text{def}}{=} \begin{cases} \Gamma(s), & \text{if } s \in \Gamma \setminus \mathrm{dom}(\Delta) \text{ or } \Gamma(s) = \Delta(s), \\ \Gamma(s) \cup \Delta(s), & \text{if } s \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) \text{ and } \Gamma(s)(a) = \Delta(s)(a), \\ \Delta(s), & \text{if } s \in \mathrm{dom}(\Delta) \setminus \mathrm{dom}(\Gamma). \end{cases}$$

Notice that this operation is not defined if $\Gamma(s)(a) \neq \Delta(s)(a)$.

The type system uses three kinds of judgments:

1. $\Gamma \vdash_s \widetilde{v{:}\tau}$, that types names, locations variables, and process variables $\tilde{v}$ at site s, with types $\tilde{\tau}$, according to the typing assumption Γ;

2. $\Gamma \vdash_s P$, saying that process P at site s conforms to the typing assumption Γ;

3. $\Gamma \vdash N$, saying that network N conforms to typing assumption Γ.

$$\text{TS-LCh}\quad \Gamma \vdash_r a@s{:}\Gamma(s)(a) \qquad \text{TS-SCh}\quad \Gamma \vdash_s a{:}\Gamma(s)(a)$$

$$\text{TS-Uni}\quad \frac{\Gamma \vdash_s \widetilde{n_1{:}\gamma_1} \quad \Delta \vdash_s \widetilde{n_2{:}\gamma_2}}{\Gamma + \Delta \vdash_s n_1\widetilde{n_2{:}\gamma_1}\gamma_1}$$

Figure 10: Typing channels and sites.

$$\text{TP-OutL}\quad \frac{\Gamma \vdash_r \widetilde{v{:}\gamma}}{\Gamma + \{s{:}\{a{:}Ch(\widetilde{\gamma})\}\} \vdash_r a@s!\langle\tilde{v}\rangle} \qquad \text{TP-OutS}\quad \frac{\Gamma \vdash_s a@s!\langle\tilde{v}\rangle}{\Gamma \vdash_s a!\langle\tilde{v}\rangle}$$

$$\text{TP-InpL}\quad \frac{\Gamma \uplus \{r{:}\{\widetilde{x{:}\gamma}\}\} \vdash_r P}{\Gamma + \{s{:}\{a{:}Ch(\widetilde{\gamma})\}\} \vdash_r a@s?(\tilde{x})P} \qquad \text{TP-InpS}\quad \frac{\Gamma \vdash_s a@s?(\tilde{x})P}{\Gamma \vdash_s a?(\tilde{x})P}$$

$$\text{TP-Par}\quad \frac{\Gamma \vdash_s P \quad \Gamma \vdash_s Q}{\Gamma \vdash_s (P \mid Q)} \qquad \text{TP-ResN}\quad \frac{\Gamma \uplus \{s{:}\varphi\} \vdash_s P}{\Gamma \vdash_s (\nu\, s)\, P}$$

$$\text{TP-ResL}\quad \frac{\Gamma \uplus \{s{:}\{a{:}\gamma\} \uplus \varphi\} \vdash_r P}{\Gamma \uplus \{s{:}\varphi\} \vdash_r (\nu\, a@s)\, P} \qquad \text{TP-ResS}\quad \frac{\Gamma \vdash_s (\nu\, a@s)\, P}{\Gamma \vdash_s (\nu\, a)\, P}$$

$$\text{TP-Weak}\quad \frac{\Gamma \vdash_s P}{\Gamma + \Delta \vdash_s P} \qquad \text{TP-Nil}\quad \emptyset \vdash_s \mathbf{0}$$

Figure 11: Typing processes.

Definition 4.3 (*lsd*π type system). The rules in Figures 10, 11, and 12 inductively define the type system of *lsd*π.

The main result is the preservation of network typability under reduction, a property usually know as *subject reduction*. The following results break the ground for it.

Definition 4.4 (Substitution on typings). 1. A *substitution of channels in a typing* is a function defined, when $x \in \mathrm{dom}(\Gamma(s))$, by the following rule:

$$\Gamma[a@r/x@s] \stackrel{\text{def}}{=} \begin{cases} \{s{:}\Gamma(s) \setminus x\} \cup \{r{:}\{a{:}\Gamma(s)(x)\} + \Gamma(r)\} \cup \Gamma \setminus s \setminus r & \text{if } r \neq s\,; \\ \{s{:}\Gamma(s) \setminus x\} + \{s{:}\{a{:}\Gamma(s)(x)\}\} \cup \Gamma \setminus s & \text{otherwise}\,. \end{cases}$$

2. A *substitution of sites in a typing* is a function defined, when $r \in \mathrm{dom}(\Gamma)$, by rule:

$$\Gamma[s/r] \stackrel{\text{def}}{=} \Gamma \setminus r + \{s{:}\Gamma(r)\}.$$

$$\text{TN-NET}\ \frac{\Gamma \vdash_s P}{\Gamma \vdash s[P]} \qquad \text{TN-NIL}\ \ \emptyset \vdash \mathbf{0} \qquad \text{TN-RESN}\ \frac{\Gamma \uplus \{s{:}\varphi\} \vdash N}{\Gamma \vdash (\nu\, s)\, N}$$

$$\text{TN-RESL}\ \frac{\Gamma \uplus \{s{:}\{a{:}\gamma\} \uplus \varphi\} \vdash N}{\Gamma \uplus \{s{:}\varphi\} \vdash (\nu\, a@s)\, N} \qquad \text{TN-PAR}\ \frac{\Gamma \vdash N \quad \Gamma \vdash M}{\Gamma \vdash (N \parallel M)}$$

Figure 12: Typing networks.

Since this definition uses the union of typings, $\Gamma[a@r/x@s]$ is not defined if $a{\in}\mathrm{dom}(\Gamma(r))$ and $\Gamma(r)(a){\neq}\Gamma(s)(x)$, and $\Gamma[s/r]$ is not defined if $s{\in}\mathrm{dom}(\Gamma)$ and $\Gamma(r){\neq}\Gamma(s)$. The following result assures that the definition of substitution of channels in a typing is correct.

Proposition 4.5 (Substitution on typings). Consider $\Delta \stackrel{\text{def}}{=} \Gamma[a@r/x@s]$ defined. Then:

1. $x \notin \mathrm{dom}(\Delta(s))$ and $\Delta \setminus s \setminus r = \Gamma \setminus s \setminus r$;
2. if $r \neq s$ then $\Delta(s) = \Gamma(s) \setminus x$, otherwise $\Delta(s) \setminus a = \Gamma(s) \setminus x$;
3. $a \in \mathrm{dom}(\Delta(r))$ and $\Delta(r)(a) = \Gamma(s)(x)$, and if $a \in \mathrm{dom}(\Gamma(r))$ then $\Gamma(r)(a) = \Gamma(s)(x)$.

Proof. Follows easily from the definitions of substitution on and union of typings. □

Since our calculus is polyadic, we are interested in simultaneous substitutions.

Notation 4.6. Let $v@s$ be $a@s$ if $v = a$, and v otherwise. Consider the *simultaneous substitution on typings* $\Gamma\{\widetilde{v@s}/\widetilde{x@s}\}$ defined similarly to the respective definition on processes (cf. Definition 3.12).

Lemma 4.7 (Simultaneous substitution). Let $\Gamma \vdash_s P$, and consider that $x \in \mathsf{fn}(P)$ but $x@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$. Then, $\Gamma\{\widetilde{v@s}/\widetilde{x@s}\} \vdash_s P\{\tilde{v}/\tilde{x}\}$.

Proof. Notice that $\Gamma\{\widetilde{v@s}/\widetilde{x@s}\}$ is only defined when $\Gamma(s)(\tilde{v}){=}\Gamma(s)(\tilde{x})$. The proof consists in an induction on the length of $\tilde{v}$, using the following auxiliary result, which is proved by induction on the derivation of the judgment $\Gamma[a@r/x@s] \vdash_s P[a@r/x]$:

Let $\Gamma \vdash_s P$, and consider that $x \in \mathsf{fn}(P)$ but $x@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$. Then,

1. $\Gamma[a@r/x@s] \vdash_s P[a@r/x]$, for all r, s; and
2. $\Gamma[a@s/x@s] \vdash_s P[a/x]$, when $r = s$.

A detailed proof can be found in page 83. □

The main result of this section states that reduction preserves the typability of a process. The lemma above is necessary when reduction results from communication. When reduction results from migration, apply the following lemma.

Lemma 4.8 (Channel translation). If $\Gamma \uplus \{s{:}\{a{:}Ch(\widetilde{\gamma})\}\} \uplus \{r{:}\{\tilde{x}{:}\widetilde{\gamma}\}\} \vdash_r P$, then $\Gamma + \{s{:}\{a{:}Ch(\widetilde{\gamma}), \tilde{x}{:}\widetilde{\gamma}\}\} \vdash_s P\sigma(\mathsf{fn}(a@s?(\tilde{x})P), r, s)$.

Proof. The proof follows on by induction on the derivation of the judgment. □

We are finally in a position to ensure the preservation of typability by reduction.

Theorem 4.9 (Subject reduction).

1. If $\Gamma \vdash_s P$ and $P \to Q$, then $\Delta \vdash_s Q$, for some Δ.
2. If $\Gamma \vdash N$ and $N \to M$, then $\Delta \vdash M$, for some Δ.

Proof. The proof consists of inductions on the derivations of $P \to Q$ and of $N \to M$. As usual, we use a lemma stating that structural congruence also preserves typability. The base case in the derivation of $P \to Q$ is when the last rule is RP-COMM. Use Lemma 4.7. There are two base cases to consider in the derivation of $N \to M$:

1. Case the last rule is RN-MIGO—apply the Definition 3.15 and the typing rules TP-OUTL and TP-OUTS subsequently.
2. Case the last rule is RN-MIGI—use then the Lemma 4.8.

The cases of the induction steps are straightforward. □

5 Comparisons and Further work

The closest calculus to *lsdπ* is Dπ. The main differences are:

1. the latter has a general migration primitive that sends arbitrary processes to remote sites, while the former only migrates messages and receptors;
2. in the latter channels are global (do not belong to a specific site), while in the former channels are local (each belong to some site, and if s and r both have a channel a, the a of s is different from the a of r);
3. in the latter sites are first-class citizens, being passed around, while in the former they are not.

We expect that the choices made in *lsdπ* do not result in loss of expressiveness, while gaining simplicity. Possible differences of expressive power in the semantics of these two calculi are under investigation. Different models of distribution provide for an explicit migration primitive; it is unclear at the time of this writing whether explicit migration may be simulated by our primitives.

We envisage to allow messages to carry sites; substitution, as defined in section 2, is ready for that. Different models of distribution provide for an explicit migration primitive; it is unclear at the time of this writing whether explicit migration may be simulated by our primitives. Finally, we would like to control unrestricted migration (cf. [8]), possibly via type systems.

Acknowledgments

This work started during a six months visit of António Ravara to the Project MIMOSA at INRIA Sophia Antipolis, France, partially supported by a post-doctoral grant of the French RNRT project MARVEL. The Portuguese PRAXIS XXI project DiCoMo was another source for financial support of the work. Thanks to Gérard Boudol, Ilaria Castellani, Matthew Hennessy, and Francisco Martins for several fruitful discussions. Thanks to Ana Matos for suggesting lsd (π).

References

[1] Roberto M. Amadio. On modelling mobility. *Theoretical Computer Science*, 240:147–176, 2000.

[2] Roberto M. Amadio, Gérard Boudol, and Cédric Lhoussaine. The receptive distributed π-calculus. Rapport de Recherche 4080, INRIA Sophia-Antipolis, 2000. A preliminary version in FST/TCS'99, LNCS 1738.

[3] Henk Barendregt. *The Lambda Calculus - Its Syntax and Semantics*. North-Holland, 1981 (1st ed.), revised 1984.

[4] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[5] Luca Cardelli. A language with distributed scope. In ACM, editor, *POPL'95: 22nd Annual ACM Symposium on Principles of Programming Languages (San Francisco, CA, U.S.A.)*, pages 286–297. ACM Press, 1995.

[6] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *Proceedings of FoSSaCS '98*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer-Verlag, 1998.

[7] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In *CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1996.

[8] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. In *HLCL'98*, volume 16 (3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 1998. Full version as CogSci Report 2/98, University of Sussex, Brighton, U. K., 1998.

[9] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-Calculus.* Cambridge University Press, 1986.

[10] Robin Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. Springer-Verlag, 1993. Available as Technical Report ECS-LFCS-91-180, University of Edinburgh, U. K., 1991.

[11] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, 1992. Available as Technical Reports ECS-LFCS-89-85 and ECS-LFCS-89-86, University of Edinburgh, U. K., 1989.

[12] Peter Sewell, Pawel Wojciechowski, and Benjamin C. Pierce. Location independence for mobile agents. In *Internet Programming Languages*, volume 1686 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[13] Vasco T. Vasconcelos and Kohei Honda. Principal typing schemes in a polyadic π-calculus. In Eike Best, editor, *Proceedings of CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 524–538. Springer-Verlag, 1993.

[14] Vasco T. Vasconcelos, Luís Lopes, and Fernando Silva. Distribution and mobility with lexical scoping in process calculi. In *HLCL'98*, volume 16 (3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 1998.

A Proofs

A.1 Results on the operational semantics

Notation A.1 (Useful sets). In this section, the following abbreviations are used with respect to the notation defined in Notation 2.2. Here, the use of channels of the form $\_@s$ is to be interpreted as "any channel located at the site s". If used inside "$\{\}$", as in $\{\_@s\}$, the resulting set is an abbreviation for $\mathcal{C}@s$. If used as an element of a set, as in $\_@s$, it represents an arbitrary element of $\mathcal{C}@s$. Furthermore, $A@s$ is not used as in Notation 2.2, but is used, instead, in place of $\mathsf{locate}(A, s)$.

Lemma A.2 (Easy auxiliary results).

1. For $\mathcal{A}_1, \ldots, \mathcal{A}_k \subseteq \mathcal{N}$, $(\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_k)[n/m] = \mathcal{A}_1[n/m] \cup \ldots \cup \mathcal{A}_k[n/m]$.
2. For X ok, if $a@t \in \mathsf{fn}(X)$ then $t \in \mathsf{fn}(X)$. Consequently, $\mathsf{sites}(\mathsf{fn}(X)) \subseteq \mathsf{fn}(X)$.
3. For X ok and $[n/m]$ a replacement, $\mathsf{fn}(X)[n/m] = \{m_1[n/m] \mid m_1 \in \mathsf{fn}(X)\}$.
4. If $\mathcal{A} \subseteq \mathcal{N}$, and Υ is a finite sequence of substitutions, then $\mathcal{A}\Upsilon = \{m_1\Upsilon \mid m_1 \in \mathcal{A}\} \cup \mathsf{sites}(\{m_1\Upsilon \mid m_1 \in \mathcal{A}\})$.

Proof of Lemma. These proofs are omitted, since they consist in straightforward applications of the definitions. $\square$

Proposition A.3 (Substitution commutes with the free names). Let Υ be a finite sequence of substitutions.

1. If Υ is a name replacement, then $\mathsf{fn}(X\Upsilon) = \mathsf{fn}(X)\Upsilon$.
2. If Υ is a name instantiation, then $\mathsf{fn}(P\Upsilon) = \mathsf{fn}(P)\Upsilon$.
3. If Υ is a name translation, then $\mathsf{fn}(P\Upsilon) \subseteq \mathsf{fn}(P)\Upsilon$.

Proof of Proposition A.3.1 and A.3.2. The proof consists of an induction over the structure of the entities. At the Process level, we provide both Propositions A.3.1 and A.3.2 at the same time. No confusion should arise from this choice, since it is clear from the context whether we are verifying a name replacement or a name instantiation. Obviously, at the Network level, it suffices to consider A.3.1.

For each case, it is necessary to perform a mathematical induction over the length of the sequence of substitutions. Therefore we will be looking at nested induction proof. We use the following abbreviations to refer to the invocation of induction base (i.b.) and hypothesis (i.h.) and distinguish between the two levels of induction.

- i.h.1, i.b.1 $\overset{\text{abv}}{=}$ i.h. and i.b. of the basic structural induction of this proof.

- i.h.2, i.b.2 $\overset{\text{abv}}{=}$ i.h. and i.b. of the secondary structural inductions of this proof.

1. Processes

   - $a@t$

     – $\mathsf{fn}(a@t[n/m]) = (\mathsf{fn}(a@t))[n/m]$, since

       * If $m = a@t$,

         $\mathsf{fn}(a@t[n/a@t])$
         $\overset{\text{def}}{=} \mathsf{fn}(n)$
         $\overset{\text{def}}{=}\{n, t\}$, since $[n/a@t]$ is a name replacement, so $n \in \{\_@t\}$;

         $(\mathsf{fn}(a@t))[n/a@t]$
         $\overset{\text{def}}{=}\{a@t, t\}[n/a@t]$
         $\overset{\text{def}}{=}\{m_1[n/a@t] \mid m_1 \in \{a@t, t\}\}\cup$
         $\cup\, \mathsf{sites}(\{m_1[n/a@t] \mid m_1 \in \{a@t, t\}\})$
         $= \{a@t[n/a@t], t[n/a@t]\} \cup \mathsf{sites}(\{a@t[n/a@t], t[n/a@t]\})$
         $\overset{\text{def}}{=}\{n, t\} \cup \mathsf{sites}(\{n, t\})$
         $\overset{\text{def}}{=}\{n, t\}$, since $[n/a@t]$ is a name replacement, so $n \in \{\_@t\}$.

       * If $m = t$,

         $\mathsf{fn}(a@t[n/t])$
         $\overset{\text{def}}{=} \mathsf{fn}(a@n) \overset{\text{def}}{=} \{a@n, n\}$

         $(\mathsf{fn}(a@t))[n/t]$
         $\overset{\text{def}}{=}\{a@t, t\}[n/t]$
         $\overset{\text{def}}{=}\{m_1[n/t] \mid m_1 \in \{a@t, t\}\} \cup \mathsf{sites}(\{m_1[n/t] \mid m_1 \in \{a@t, t\}\})$
         $= \{a@t[n/t], t[n/t]\} \cup \mathsf{sites}(\{a@t[n/t], t[n/t]\})$
         $\overset{\text{def}}{=}\{a@n, n\} \cup \mathsf{sites}(\{a@n, n\})$
         $\overset{\text{def}}{=}\{a@n, n\}$.

       * If $t \notin m$,

         $\mathsf{fn}(a@t[n/m])$
         $\overset{\text{def}}{=} \mathsf{fn}(a@t)$
         $\overset{\text{def}}{=}\{a@t, t\}$;

$(\mathsf{fn}(a@t))[n/m]$
$\overset{\text{def}}{=}\{a@t, t\}[n/m]$
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \{a@t, t\}\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \{a@t, t\}\})$
$= \{a@t[n/m], t[n/m]\} \cup \mathsf{sites}(\{a@t[n/m], t[n/m]\})$
$\overset{\text{def}}{=}\{a@t, t\} \cup \mathsf{sites}(\{a@t, t\})$
$\overset{\text{def}}{=}\{a@t, t\}.$

- $\mathsf{fn}(a@t[n_1/m_1]\ldots[n_f/m_f])$
  By definition, $a@t[n_1/m_1] \in \mathcal{C}@\mathcal{S}$,
  $=$ (i.h.2) $\mathsf{fn}(a@t[n_1/m_1])[n_2/m_2]\ldots[n_f/m_f]$
  $=$ (i.b.2) $(\mathsf{fn}(a@t))[n_1/m_1]\ldots[n_f/m_f]$

- a
  - $\mathsf{fn}(a[n/m]) = (\mathsf{fn}(a))[n/m]$, for

    * If $m = a$,

      $\mathsf{fn}(a[n/a])$
      $\overset{\text{def}}{=}\mathsf{fn}(n)$
      $\overset{\text{def}}{=}\{n\} \cup \mathsf{sites}(\{n\})$;

      $(\mathsf{fn}(a))[n/a]$
      $\overset{\text{def}}{=}\{m_1[n/a] \mid m_1 \in \mathsf{fn}(a)\} \cup \mathsf{sites}(\{m_1[n/a] \mid m_1 \in \mathsf{fn}(a)\})$
      $\overset{\text{def}}{=}\{a[n/a]\} \cup \mathsf{sites}(\{a[n/a]\})$
      $\overset{\text{def}}{=}\{n\} \cup \mathsf{sites}(\{n\})$.

    * If $m \neq a$,

      $\mathsf{fn}(a[n/m])$
      $\overset{\text{def}}{=}\mathsf{fn}(a)$
      $\overset{\text{def}}{=}\{a\}$;

      $(\mathsf{fn}(a))[n/m]$
      $\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \mathsf{fn}(a)\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(a)\})$
      $\overset{\text{def}}{=}\{a[n/m]\} \cup \mathsf{sites}(\{a[n/m]\})$
      $\overset{\text{def}}{=}\{a\}$.

  - $\mathsf{fn}(a[n_1/m_1]\ldots[n_f/m_f])$
    By definition, $a[n_1/m_1] \in \mathit{Chans}$. If $a[n_1/m_1] \in \mathcal{C}$ use i.h.2, and if $a[n_1/m_1] \in$

$\mathcal{C}@\mathcal{S}$ use the previous case.

$= \mathsf{fn}(a[n_1/m_1])[n_2/m_2]\dots[n_f/m_f]$
$=$ (i.b.2) $(\mathsf{fn}(a))[n_1/m_1]\dots[n_f/m_f]$.

- s
  - $\mathsf{fn}(s[n/m]) = (\mathsf{fn}(s))[n/m]$, for
    - If $m = s$,

      $\mathsf{fn}(s[n/s])$
      $\overset{\text{def}}{=} \mathsf{fn}(n)$
      $\overset{\text{def}}{=} \{n\}$, since $[n/s]$ is a name replacement, then $n \in \mathcal{S}$;

      $(\mathsf{fn}(s))[n/s]$
      $\overset{\text{def}}{=} \{m_1[n/s] \mid m_1 \in \mathsf{fn}(s)\} \cup \mathsf{sites}(\{m_1[n/s] \mid m_1 \in \mathsf{fn}(s)\})$
      $\overset{\text{def}}{=} \{s[n/s]\} \cup \mathsf{sites}(\{s[n/s]\})$
      $\overset{\text{def}}{=} \{n\} \cup \mathsf{sites}(\{n\})$
      $\overset{\text{def}}{=} \{n\}$, since $[n/s]$ is a name replacement, then $n \in \mathcal{S}$.

    - If $m \neq s$,

      $\mathsf{fn}(s[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}(s)$
      $\overset{\text{def}}{=} \{s\}$;

      $(\mathsf{fn}(s))[n/m]$
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(s)\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(s)\})$
      $\overset{\text{def}}{=} \{s[n/m]\} \cup \mathsf{sites}(\{s[n/m]\})$
      $\overset{\text{def}}{=} \{s\}$.

  - $\mathsf{fn}(s[n_1/m_1]\dots[n_f/m_f])$.
    By definition $s[n_1/m_1] \in \mathcal{S}$,
    $=$ (i.h.2) $\mathsf{fn}(s[n_1/m_1])[n_2/m_2]\dots[n_f/m_f]$
    $=$ (i.b.2) $(\mathsf{fn}(s))[n_1/m_1]\dots[n_f/m_f]$.

- $\mathbf{0}$

- $\mathsf{fn}(\mathbf{0}[n/m]) = (\mathsf{fn}(\mathbf{0}))[n/m]$, for

  $\mathsf{fn}(\mathbf{0}[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(\mathbf{0})$
  $\overset{\text{def}}{=} \emptyset$;

  $(\mathsf{fn}(\mathbf{0}))[n/m]$
  $\overset{\text{def}}{=} \emptyset[n/m]$
  $= \emptyset$.

- $\mathsf{fn}(\mathbf{0}[n_1/m_1]\dots[n_f/m_f])$
  $\overset{\text{def}}{=} \mathsf{fn}(\mathbf{0}[n_2/m_2]\dots[n_f/m_f])$
  $=$ (i.h.2) $\mathsf{fn}(\mathbf{0})[n_2/m_2]\dots[n_f/m_f]$
  $=$ (i.b.2) $\mathsf{fn}(\mathbf{0})[n_1/m_1]\dots[n_f/m_f]$

• $u!\langle\widetilde{n}\rangle$

- $\mathsf{fn}(u!\langle\widetilde{n}\rangle[n/m]) = (\mathsf{fn}(u!\langle\widetilde{n}\rangle))[n/m]$, for

  $\mathsf{fn}((u!\langle\widetilde{n}\rangle)[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]!\langle n_1[n/m]\dots n_f[n/m]\rangle)$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]) \cup \mathsf{fn}(n_1[n/m]) \cup \dots \cup \mathsf{fn}(n_f[n/m])$
  $=$ (i.h.1) $\mathsf{fn}(u)[n/m] \cup \mathsf{fn}(n_1[n/m]) \cup \dots \cup \mathsf{fn}(n_f)[n/m]$;

  $(\mathsf{fn}(u!\langle\widetilde{n}\rangle))[n/m]$
  $\overset{\text{def}}{=} (\mathsf{fn}(u) \cup \mathsf{fn}(n_1) \cup \dots \cup \mathsf{fn}(n_f))[n/m]$
  $=$ (Lemma A.2.1) $\mathsf{fn}(u)[n/m] \cup \mathsf{fn}(n_1)[n/m] \cup \dots \cup \mathsf{fn}(n_f)[n/m]$.

- $\mathsf{fn}(u!\langle\widetilde{n}\rangle[n_1/m_1]\dots[n_f/m_f])$.
  Since by definition $u!\langle\widetilde{n}\rangle[n_1/m_1]$ is of the form $u'!\langle n'n'\rangle$,
  $=$ (i.h.2) $\mathsf{fn}(u!\langle\widetilde{n}\rangle[n_1/m_1])[n_2/m_2]\dots[n_f/m_f]$
  $=$ (i.b.2) $(\mathsf{fn}(u!\langle\widetilde{n}\rangle))[n_1/m_1]\dots[n_f/m_f]$.

• $P \mid Q$

- $\mathsf{fn}((P \mid Q)[n/m]) = (\mathsf{fn}(P \mid Q))[n/m]$, for

  $\mathsf{fn}((P \mid Q)[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}((P)[n/m] \mid (Q)[n/m])$

$\overset{\text{def}}{=} \mathsf{fn}((P)[n/m]) \cup \mathsf{fn}((Q)[n/m])$
$=$ (i.h.1) $\mathsf{fn}(P)[n/m] \cup \mathsf{fn}(Q)[n/m]$.

$(\mathsf{fn}(P \mid Q))[n/m]$
$\overset{\text{def}}{=} (\mathsf{fn}(P) \cup \mathsf{fn}(Q))[n/m]$
$=$ (Lemma A.2.1) $\mathsf{fn}(P)[n/m] \cup \mathsf{fn}(Q)[n/m]$.

- $\mathsf{fn}((P \mid Q)[n_1/m_1] \dots [n_f/m_f])$.
  Since by definition $(P \mid Q)[n_1/m_1]$ is of the form $P' \mid Q'$,
  $=$ (i.h.2) $\mathsf{fn}(P[n_1/m_1] \mid Q[n_1/m_1])[n_2/m_2] \dots [n_f/m_f]$
  $=$ (i.b.2) $(\mathsf{fn}(P \mid Q))[n_1/m_1] \dots [n_f/m_f]$.

- $(\nu\, t)\, Q$

  - $\mathsf{fn}((\nu\, t)\, Q[n/m]) = (\mathsf{fn}((\nu\, t)\, Q))[n/m]$, for

    - If $t \in m$,

      $\mathsf{fn}(((\nu\, t)\, Q)[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}((\nu\, t)\, Q)$
      $\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{\_@t, t\}$

      $(\mathsf{fn}((\nu\, t)\, Q))[n/m]$
      $\overset{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{\_@t, t\})[n/m]$
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\} \cup$
      $\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\})$
      $\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{\_@t, t\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{\_@t, t\})$,
      because $t \notin m_1$ but $t \in m$
      $=$ (Lemma A.2.2) $\mathsf{fn}(Q) \setminus \{\_@t, t\}$

    - If $t \notin m$, and $t \notin n$ or $m \notin \mathsf{fn}(Q)$

      $\mathsf{fn}(((\nu\, t)\, Q)[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}((\nu\, t)\, Q[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}(Q[n/m]) \setminus \{\_@t, t\}$

      $(\mathsf{fn}((\nu\, t)\, Q))[n/m]$
      $\overset{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{\_@t, t\})[n/m]$
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\} \cup$
      $\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\})$

By hypothesis,

$= \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\_@t, t\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{\_@t, t\}$
$\overset{\text{def}}{=} (\mathsf{fn}(Q)[n/m]) \setminus \{\_@t, t\}$
$=$ (i.h.1) $\mathsf{fn}(Q[n/m]) \setminus \{\_@t, t\}$

$*$ If $t \notin m$, and $t \in n$ and $m \in \mathsf{fn}(Q)$, where s is fresh.

$\mathsf{fn}(((\nu\, t)\, Q)[n/m])$
$\overset{\text{def}}{=} \mathsf{fn}((\nu\, s)\, Q[s/t][n/m])$
$\overset{\text{def}}{=} \mathsf{fn}(Q[s/t][n/m]) \setminus \{\_@s, s\}$

$(\mathsf{fn}((\nu\, t)\, Q))[n/m]$
$\overset{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{\_@t, t\})[n/m]$
$\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\_@t, t\}\})$
$=$ (*) $\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[s/t]) \setminus \{\_@s, s\}\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[s/t]) \setminus \{\_@s, s\}\})$

Since s is fresh, $s \notin n, m$, thus
$= \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[s/t]\} \setminus \{\_@s, s\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[s/t]\}) \setminus \{\_@s, s\}$
$\overset{\text{def}}{=} (\mathsf{fn}(Q)[s/t][n/m]) \setminus \{\_@s, s\}$
$=$ (i.h.1) $\mathsf{fn}(Q[s/t][n/m]) \setminus \{\_@s, s\}$ (i.h.1. may be used since $[s/t]$ is a name replacement)

(*) $((\mathsf{fn}(Q)[s/t]) \setminus \{\_@s, s\}$
$\overset{\text{def}}{=} \{m_1[s/t] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\_@s, s\} \cup \mathsf{sites}(\{m_1[s/t] \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{\_@s, s\}$
$\overset{\text{def}}{=} \{m_1 \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\_@t, t\} \cup \mathsf{sites}(\{m_1 \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{\_@t, t\}$
$= \mathsf{fn}(Q) \setminus \{\_@t, t\}$

– $\mathsf{fn}(((\nu\, t)\, Q)[n_1/m_1] \ldots [n_f/m_f])$

By definition $((\nu\, t)\, Q)[n_1/m_1]$ is of the form $(\nu\, t')\, Q'$

$=$ (i.h.2) $\mathsf{fn}(((\nu\, t)\, Q)[n_1/m_1])[n_2/m_2] \ldots [n_f/m_f]$
$=$ (i.b.2) $(\mathsf{fn}((\nu\, t)\, Q))[n_1/m_1] \ldots [n_f/m_f]$

- $(\nu\, a)\, Q$
  - $\mathsf{fn}((\nu\, a)\, Q[n/m]) = (\mathsf{fn}((\nu\, a)\, Q))[n/m]$, since
    * If $m = a$,

      $\mathsf{fn}(((\nu\, a)\, Q)[n/a])$
      $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a)\, Q)$
      $\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{a\}$

      $(\mathsf{fn}((\nu\, a)\, Q))[n/a]$
      $\overset{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{a\})[n/a]$
      $\overset{\text{def}}{=} \{m_1[n/a] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\} \cup \mathsf{sites}(\{m_1[n/a] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\})$
      $\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{a\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{a\})$, since $m_1 \neq a$
      $=$ (Lemma A.2.2) $\mathsf{fn}(Q) \setminus \{a\}$

    * If $m \neq a$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$

      $\mathsf{fn}(((\nu\, a)\, Q)[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a)\, Q[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}(Q[n/m]) \setminus \{a\}$

      $(\mathsf{fn}((\nu\, a)\, Q))[n/m]$
      $\overset{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{a\})[n/m]$
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\})$

      By hypothesis,

      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a\})$
      $\overset{\text{def}}{=} (\mathsf{fn}(Q)[n/m]) \setminus \{a\}$
      $=$ (i.h.1) $\mathsf{fn}(Q[n/m]) \setminus \{a\}$

    * If $m \neq a$, and $a \in n$ and $m \in \mathsf{fn}(Q)$, where b is fresh,

      $\mathsf{fn}(((\nu\, a)\, Q)[n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}((\nu\, b)\, Q[b/a][n/m])$
      $\overset{\text{def}}{=} \mathsf{fn}(Q[b/a][n/m]) \setminus \{b\}$

      $(\mathsf{fn}((\nu\, a)\, Q))[n/m]$

$\overset{\text{def}}{=}(\mathsf{fn}(Q) \setminus \{a\})[n/m]$
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a\}\})$
$= (*)\ \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b/a]\{b\}\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b/a]\{b\}\})$

Since b is fresh, $b \notin n, m$, so
$= \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b/a]\} \setminus \{b\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b/a]\}) \setminus \{b\}$

$\overset{\text{def}}{=}(\mathsf{fn}(Q)[b/a][n/m]) \setminus \{b\}$
$=$ (i.h.1) $\mathsf{fn}(Q[b/a][n/m]) \setminus \{b\}$ (i.h.1. may be applied because $[b/a]$ is replacement.)

(*) $((\mathsf{fn}(Q)[b/a]) \setminus \{b\}$
$\overset{\text{def}}{=}\{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{b\} \cup \mathsf{sites}(\{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{b\}$
$\overset{\text{def}}{=}\{m_1 \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a\} \cup \mathsf{sites}(\{m_1 \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{a\}$
$= \mathsf{fn}(Q) \setminus \{a\}$

– $\mathsf{fn}(((\nu\, a)\, Q)[n_1/m_1] \ldots [n_f/m_f])$

Since by definition $((\nu\, a)\, Q)[n_1/m_1]$ is of the form $(\nu\, a')\, Q'$

$=$ (i.h.2) $\mathsf{fn}((((\nu\, a)\, Q)[n_1/m_1])[n_2/m_2] \ldots [n_f/m_f]$
$=$ (i.b.2) $(\mathsf{fn}((\nu\, a)\, Q))[n_1/m_1] \ldots [n_f/m_f]$

- $(\nu\, a@t)\, Q$

  – $\mathsf{fn}((\nu\, a@t)\, Q[n/m]) = (\mathsf{fn}((\nu\, a@t)\, Q))[n/m]$, for

  ∗ If $m = a@t$,

  $\mathsf{fn}(((\nu\, a@t)\, Q)[n/a@t])$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a@t)\, Q)$
  $\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\}$

  $(\mathsf{fn}((\nu\, a@t)\, Q))[n/a@t]$
  $\overset{\text{def}}{=}(\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})[n/a@t]$
  $\overset{\text{def}}{=}\{m_1[n/a@t] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\}\cup$
  $\cup\, \mathsf{sites}(\{m_1[n/a@t] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\})$

$\stackrel{\text{def}}{=} \mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})$, since $m_1 \neq a@t$ but $m = a@t$
$=$ (Lemma A.2.2) $\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\}$

$*$ If $m = t$,

$\mathsf{fn}(((\nu\, a@t)\, Q)[n/t])$
$\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a@n)\, Q[n/t])$
$\stackrel{\text{def}}{=} \mathsf{fn}(Q[n/t]) \setminus \{a@n\} \cup \{n\}$

$(\mathsf{fn}((\nu\, a@t)\, Q))[n/t]$
$\stackrel{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})[n/t]$
$\stackrel{\text{def}}{=} \{m_1[n/t] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/t] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\})$
$\stackrel{\text{def}}{=} \{m_1[n/t] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a@t\}\} \cup \{n\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/t] \mid m_1 \in \mathsf{fn}(Q) \setminus \{a@t\}\} \cup \{n\})$
$\stackrel{\text{def}}{=} \{m_1 \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a@n\} \cup \{n\} \cup \mathsf{sites}(\{m_1 \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a@t\} \cup \{n\})$
$= (\mathsf{fn}(Q)[n/t]) \setminus \{a@n\} \cup \{n\} \cup \mathsf{sites}((\mathsf{fn}(Q)[n/t]) \setminus \{a@n\} \cup \{n\})$
$=$ (i.h.1) $\mathsf{fn}(Q[n/t]) \setminus \{a@n\} \cup \{n\} \cup \mathsf{sites}(\mathsf{fn}(Q[n/t]) \setminus \{a@n\})$
$=$ (Lemma A.2.2) $\mathsf{fn}(Q[n/t]) \setminus \{a@n\} \cup \{n\}$

$*$ If $m \neq a@t, t$, and $n \neq a$,$a@t$ or $m \notin \mathsf{fn}(M)$

$\mathsf{fn}(((\nu\, a@t)\, Q)[n/m])$
$\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a@t)\, Q[n/m])$
$\stackrel{\text{def}}{=} \mathsf{fn}(Q[n/m]) \setminus \{a@t\} \cup \{t\}$

$(\mathsf{fn}((\nu\, a@t)\, Q))[n/m]$
$\stackrel{\text{def}}{=} (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})[n/m]$
$\stackrel{\text{def}}{=} \{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\})$

By hypothesis,

$\stackrel{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a@t\} \cup \{t\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a@t\} \cup \{t\})$
$= (\mathsf{fn}(Q)[n/m]) \setminus \{a@t\} \cup \{t\}$
$=$ (i.h.1) $\mathsf{fn}(Q[n/m]) \setminus \{a@t\} \cup \{t\}$

$*$ If $m \neq a@t$ and $m \neq t$, and $n \in \{a, a@t\}$ and $m \in \mathsf{fn}(M)$, where b is fresh

$\mathsf{fn}(((\nu\, a@t)\, Q)[n/m])$
$\overset{\text{def}}{=} \mathsf{fn}((\nu\, b@t)\, M[b@t/a@t][n/m])$
$\overset{\text{def}}{=} \mathsf{fn}(M[b@t/a@t][n/m]) \setminus \{b@t\} \cup \{t\}$

$(\mathsf{fn}((\nu\, a@t)\, Q))[n/m]$
$\overset{\text{def}}{=}(\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})[n/m]$
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\})\})$
$=$ (*) $\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b@t/a@t]) \setminus \{b@t\}\cup \cup\{t\}\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b@t/a@t]) \setminus \{b@t\} \cup \{t\}\})$

Since b is fresh, $b \notin n, m$, and since $m \neq t$

$= \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b@t/a@t]\} \setminus \{b@t\} \cup \{t\}\cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)[b@t/a@t]\} \setminus \{b@t\} \cup \{t\})$
$\overset{\text{def}}{=}(\mathsf{fn}(Q)[b@t/a@t][n/m]) \setminus \{b@t\} \cup \{t\}$
$=$ (i.h.1) $\mathsf{fn}(Q[b@t/a@t][n/m]) \setminus \{b@t\} \cup \{t\}$ (i.h.1. may be used because $[b@t/a@t]$ is a name replacement)

(*) $((\mathsf{fn}(Q)[b@t/a@t]) \setminus \{b@t\} \cup \{t\}$
$\overset{\text{def}}{=}\{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{b@t\} \cup \{t\}\cup \cup$
$ts\{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{b@t\} \cup \{t\}$

Since b is fresh, $b@t \notin \mathsf{fn}(Q)$,

$\overset{\text{def}}{=}\{m_1 \mid m_1 \in \mathsf{fn}(Q)\}\setminus\{a@t\}\cup\{t\}\cup\mathsf{sites}(\{m_1 \mid m_1 \in \mathsf{fn}(Q)\})\setminus\{a@t\}\cup\{t\}$
$= \mathsf{fn}(Q) \setminus \{a@t\} \cup \{t\}$

- $\mathsf{fn}(((\nu\, a@t)\, Q)[n_1/m_1]\ldots[n_f/m_f])$

  Since by definition $((\nu\, a@t)\, Q)[n_1/m_1]$ is of the form $(\nu\, a'@t)\, Q'$

  $=$ (i.h.2) $\mathsf{fn}(((\nu\, a@t)\, Q)[n_1/m_1])[n_2/m_2]\ldots[n_f/m_f]$
  $=$ (i.b.2) $(\mathsf{fn}((\nu\, a@t)\, Q))[n_1/m_1]\ldots[n_f/m_f]$

- $u?(\tilde{a})Q$

  - $\mathsf{fn}(u?(\tilde{a})Q[n/m]) = (\mathsf{fn}(u?(\tilde{a})Q))[n/m]$, for

    - If $m = a_i$ and $a_i \in \{\tilde{a}\}$,

$\mathsf{fn}((u?(\tilde{a})Q)[n/a_i])$
$\overset{\text{def}}{=} \mathsf{fn}(u[n/a_i]?(\tilde{a})Q)$
$\overset{\text{def}}{=} \mathsf{fn}(Q) \setminus \{\tilde{a}\} \cup \{u\} \cup \mathsf{fn}(u)$

$(\mathsf{fn}(u?(\tilde{a})Q))[n/a_i]$
$\overset{\text{def}}{=} (\mathsf{fn}(u) \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/a_i]$
$=$ (Lemma A.2.1) $\mathsf{fn}(u)[n/m] \cup (\mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/a_i]$
$\overset{\text{def}}{=} \mathsf{fn}(u)[n/a_i] \cup \{m_1[n/a_i] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/a_i] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\})$
$\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{\tilde{a}\})$ since $m_1 \notin \{\tilde{a}\}$
$=$ (Lemma A.2.2) $\mathsf{fn}(u)[n/m] \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\}$
$=$ (i.h.1) $\mathsf{fn}(u[n/m]) \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\}$

* If $m \notin \{\tilde{a}\}$, and $\forall a_i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$

  $\mathsf{fn}((u?(\tilde{a})Q)[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]?(\tilde{a})Q[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]) \cup \mathsf{fn}(Q[n/m]) \setminus \{\tilde{a}\}$

  $\mathsf{fn}((u?(\tilde{a})Q))[n/m]$
  $\overset{\text{def}}{=} (\mathsf{fn}(u) \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/m]$
  $=$ (Lemma A.2.1) $\mathsf{fn}(u)[n/m] \cup (\mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/m]$
  $\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\} \cup$
  $\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\})$

  By hypothesis,

  $\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\tilde{a}\} \cup$
  $\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\tilde{a}\}\})$
  $\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup (\mathsf{fn}(Q)[n/m]) \setminus \{\tilde{a}\}$
  $=$ (i.h.1) $\mathsf{fn}(u[n/m]) \cup \mathsf{fn}(Q[n/m]) \setminus \{\tilde{a}\}$

* If $m \notin \{\tilde{a}\}$, and for some $i : a_i \in n$ and $m \in \mathsf{fn}(Q)$

  $\mathsf{fn}((u?(\tilde{a})Q)[n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]?(a_1 \ldots b \ldots a_n)Q[b/a_i][n/m])$
  $\overset{\text{def}}{=} \mathsf{fn}(u[n/m]) \cup \mathsf{fn}(Q[b/a_i][n/m]) \setminus \{a_1 \ldots b \ldots a_n\}$

  $\mathsf{fn}((u?(\tilde{a})Q))[n/m]$

$\overset{\text{def}}{=}(\mathsf{fn}(u) \cup \mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/m]$
$=$ (Lemma A.2.1) $\mathsf{fn}(u)[n/m] \cup (\mathsf{fn}(Q) \setminus \{\tilde{a}\})[n/m]$
$\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q) \setminus \{\tilde{a}\}\})$
$=$ (*) $\mathsf{fn}(u)[n/m] \cup \{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b/a_i]) \setminus \{a_1 \dots b \dots a_n\}\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b/a_i]) \setminus \{a_1 \dots b \dots a_n\}\})$

Since b is fresh, $b \notin n, m$, so
$= \mathsf{fn}(u)[n/m] \cup \{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b/a_i])\} \setminus \{a_1 \dots b \dots a_n\} \cup$
$\cup\, \mathsf{sites}(\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)[b/a_i])\} \setminus \{a_1 \dots b \dots a_n\})$

$\overset{\text{def}}{=} \mathsf{fn}(u)[n/m] \cup (\mathsf{fn}(Q)[b/a_i][n/m]) \setminus \{a_1 \dots b \dots a_n\}$
$=$ (i.h.1) $\mathsf{fn}(u)[n/m] \cup \mathsf{fn}(Q[b/a_i][n/m]) \setminus \{a_1 \dots b \dots a_n\}$ (i.h.1. may be used because $[b/a_i]$ is a name replacement)

(*) $((\mathsf{fn}(Q)[b/a_i]) \setminus \{a_1 \dots b \dots a_n\}$
$\overset{\text{def}}{=}\{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{a_1 \dots b \dots a_n\} \cup$
$\cup\, \mathsf{sites}(\{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{a_1 \dots b \dots a_n\}$
$\overset{\text{def}}{=}\{m_1 \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{\tilde{a}\} \cup$
$\cup\, \mathsf{sites}(\{m_1 \mid m_1 \in \mathsf{fn}(Q)\}) \setminus \{\tilde{a}\}$
$=$ (Lemma A.2.2) $\mathsf{fn}(Q) \setminus \{\tilde{a}\}$

- $\mathsf{fn}((u?(\tilde{a})Q)[n_1/m_1] \dots [n_f/m_f])$

  Since by definition $(u?(\tilde{a})Q)[n_1/m_1]$ is of the form $u'?(\tilde{a'})Q'$,

  $=$ (i.h.2) $\mathsf{fn}((u?(\tilde{a})Q)[n_1/m_1])[n_2/m_2] \dots [n_f/m_f]$
  $=$ (i.b.2) $(\mathsf{fn}(u?(\tilde{a})Q))[n_1/m_1] \dots [n_f/m_f]$

- Networks

Note that, as mentioned in the beginning of the proof, from now on we consider name replacement alone.

- $s[Q]$

  - $\mathsf{fn}(s[Q][n/m]) = (\mathsf{fn}(s[Q]))[n/m]$, for

    * If $m = a@s$,

      $\mathsf{fn}((s[Q])[b@s/a@s])$

$\overset{\text{def}}{=} \mathsf{fn}(s[(Q[b/a])[b@s/a@s]])$
$\overset{\text{def}}{=}\mathsf{fn}((Q[b/a])[b@s/a@s])@s \cup \{s\}$

$\mathsf{fn}(s[Q])[b@s/a@s]$
$\overset{\text{def}}{=}(\mathsf{fn}(Q)@s \cup \{s\})[b@s/a@s]$
$\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[b@s/a@s] \mid m_1 \in (\mathsf{fn}(Q)@s \cup \{s\})\}$
$=$ $\{m_1[b/a][b@s/a@s] \mid m_1 \in \mathsf{fn}(Q)\}@s \cup \{s\}$ (the substitution doesn't change s)
$\overset{\text{def}}{=}(\mathsf{fn}(Q)[b/a][b@s/a@s])@s \cup \{s\}$
$=$ (i.h.1) $\mathsf{fn}(Q[b/a][b@s/a@s])@s \cup \{s\}$ (i.h.1. may be used because $[b/a]$ and $[b@s/a@s]$ are name replacements)

∗ If $m = s$,

$\mathsf{fn}((s[Q])[n/s])$
$\overset{\text{def}}{=} \mathsf{fn}(n[Q[n/s]])$
$\overset{\text{def}}{=}\mathsf{fn}(Q[n/s])@n \cup \{n\}$

$\mathsf{fn}(s[Q])[n/s]$
$\overset{\text{def}}{=}(\mathsf{fn}(Q)@s \cup \{s\})[n/s]$
$\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[n/s] \mid m_1 \in (\mathsf{fn}(Q)@s \cup \{s\})\}$
$= \{m_1[n/s] \mid m_1 \in \mathsf{fn}(Q)\}@n \cup \{n\}$
$\overset{\text{def}}{=}\mathsf{fn}(Q)[n/s]@n \cup \{n\}$
$=$ (i.h.1) $\mathsf{fn}(Q[n/s])@n \cup \{n\}$

∗ If $s \notin m$,

$\mathsf{fn}((s[Q])[n/m])$
$\overset{\text{def}}{=} \mathsf{fn}(s[P[n/m]])$
$\overset{\text{def}}{=}\mathsf{fn}(P[n/m])@s \cup \{s\}$

$\mathsf{fn}(s[Q])[n/m]$
$\overset{\text{def}}{=}(\mathsf{fn}(Q)@s \cup \{s\})[n/m]$
$\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[n/m] \mid m_1 \in (\mathsf{fn}(Q)@s \cup \{s\})\}$
$= \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}@s \cup \{s\}$
$\overset{\text{def}}{=}(\mathsf{fn}(Q)[n/m])@s \cup \{s\}$
$=$ (i.h.1) $\mathsf{fn}(Q[n/m])@s \cup \{s\}$

– $\mathsf{fn}((s[Q])[n_1/m_1] \dots [n_f/m_f])$

Since by definition $(s[Q])[n_1/m_1]$ is of the form $s'[Q']$ then

$= \text{(i.h.2)}\ \mathsf{fn}((s[Q])[n_1/m_1])[n_2/m_2]\dots[n_f/m_f]$
$= \text{(i.b.2)}\ (\mathsf{fn}(s[Q]))[n_1/m_1]\dots[n_f/m_f]$

- $\mathbf{0}$, $M_1 \parallel M_2$, $(\nu\, a@t)\, M$

  As in the case for processes.

$\square$

Proof of Proposition A.3.3. This proof is analogous to the previous one. Note that, by definition of $X\ \ \mathsf{ok}$, the i.h. may be applied to all subterms of X.

1. Processes

   - a

     – $\mathsf{fn}(a[u/v]) = (\mathsf{fn}(a))[u/v]$, for

       ∗ If $v = a$,

         $\mathsf{fn}(a[u/a])$
         $\overset{\text{def}}{=} \mathsf{fn}(u)$
         $\overset{\text{def}}{=} \{u\} \cup \mathsf{sites}(\{u\})$

         $(\mathsf{fn}(a))[u/a]$
         $\overset{\text{def}}{=} \{m_1[u/a] \mid m_1 \in \mathsf{fn}(a)\} \cup \mathsf{sites}(\{m_1[u/a] \mid m_1 \in \mathsf{fn}(a)\})$
         $\overset{\text{def}}{=} \{a[u/a]\} \cup \mathsf{sites}(\{a[u/a]\})$
         $\overset{\text{def}}{=} \{u\} \cup \mathsf{sites}(\{u\})$

       ∗ If $v \neq a$,

         $\mathsf{fn}(a[u/v])$
         $\overset{\text{def}}{=} \mathsf{fn}(a)$
         $\overset{\text{def}}{=} \{a\}$

         $(\mathsf{fn}(a))[u/v]$
         $\overset{\text{def}}{=} \{m_1[u/v] \mid m_1 \in \mathsf{fn}(a)\} \cup \mathsf{sites}(\{m_1[u/v] \mid m_1 \in \mathsf{fn}(a)\})$

$\overset{\text{def}}{=}\{a[u/v]\} \cup \mathsf{sites}(\{a[u/v]\})$ since $v \neq a$
$\overset{\text{def}}{=}\{a\}$

    * $\mathsf{fn}(a[u_1/v_1]\dots[u_f/v_f])$
      $=$ (i.h.2) $\mathsf{fn}(a[u_1/v_1])[u_2/v_2]\dots[u_f/v_f]$
      $=$ (i.b.2) $(\mathsf{fn}(a))[u_1/v_1]\dots[u_f/v_f]$

- s

  - $\mathsf{fn}(s[u/v]) = (\mathsf{fn}(s))[u/v]$, for

    * Of course that $v \neq s$,

      $\mathsf{fn}(s[u/v])$
      $\overset{\text{def}}{=}\mathsf{fn}(s)$
      $\overset{\text{def}}{=}\{s\}$

      $(\mathsf{fn}(s))[u/v]$
      $\overset{\text{def}}{=}\{m_1[u/v] \mid m_1 \in \mathsf{fn}(s)\} \cup \mathsf{sites}(\{m_1[u/v] \mid m_1 \in \mathsf{fn}(s)\})$
      $\overset{\text{def}}{=}\{s[u/v]\} \cup \mathsf{sites}(\{s[u/v]\})$
      $\overset{\text{def}}{=}\{s\}$

  - $\mathsf{fn}(s[u_1/v_1]\dots[u_f/v_f])$
    $=$ (i.h.2) $\mathsf{fn}(s[u_1/v_1])[u_2/v_2]\dots[u_f/v_f]$
    $=$ (i.b.2) $(\mathsf{fn}(s))[u_1/v_1]\dots[u_f/v_f]$

- $a@t$

  - $\mathsf{fn}(a@t[u/v]) = (\mathsf{fn}(a@t))[u/v]$, for

    * If $v = a@t$, then $u = a$, since $[u/v]$ is a name translation

      $\mathsf{fn}(a@t[a/a@t])$
      $\overset{\text{def}}{=}\mathsf{fn}(a)$
      $\overset{\text{def}}{=}\{a\}$

      $(\mathsf{fn}(a@t))[a/a@t]$
      $\overset{\text{def}}{=}\{a@t, t\}[a/a@t]$
      $\overset{\text{def}}{=}\{m_1[a/a@t] \mid m_1 \in \{a@t, t\}\} \cup \mathsf{sites}(\{m_1[a/a@t] \mid m_1 \in \{a@t, t\}\})$
      $= \{a@t[a/a@t], t[a/a@t]\} \cup \mathsf{sites}(\{a@t[a/a@t], t[a/a@t]\})$

$\overset{\text{def}}{=}\{a,t\} \cup \mathsf{sites}(\{a,t\})$
$\overset{\text{def}}{=}\{a,t\}$

$*$ If $t \notin v$,

$\mathsf{fn}(a@t[u/v])$
$\overset{\text{def}}{=} \mathsf{fn}(a@t)$
$\overset{\text{def}}{=}\{a@t,t\}$

$(\mathsf{fn}(a@t))[u/v]$
$\overset{\text{def}}{=}\{a@t,t\}[u/v]$
$\overset{\text{def}}{=}\{m_1[u/v] \mid m_1 \in \{a@t,t\}\} \cup \mathsf{sites}(\{m_1[u/v] \mid m_1 \in \{a@t,t\}\})$
$= \{a@t[u/v], t[u/v]\} \cup \mathsf{sites}(\{a@t[u/m], t[u/v]\})$
$\overset{\text{def}}{=}\{a@t,t\} \cup \mathsf{sites}(\{a@t,t\})$ since $t \notin v$,
$\overset{\text{def}}{=}\{a@t,t\}$

– $\mathsf{fn}(a@t[u_1/v_1]\dots[u_f/v_f])$
$=$ (i.h.2) $\mathsf{fn}(a@t[u_1/v_1])[u_2/v_2]\dots[u_f/v_f]$
$=$ (i.b.2) $(\mathsf{fn}(a@t))[u_1/v_1]\dots[u_f/v_f]$

2. Networks

Note that the rest is analogous to Proof A.3.1, but u,v is used instead of m,n, and in place of the equality invoked by i.h.1, the $\subseteq$ relation is used.

$\square$

Proposition A.4 (Substitution preserves the syntactic restrictions). Let Υ be a finite sequence of substitutions.

1. If X ok and Υ is a name replacement, then $X\Upsilon$ ok.
2. If P ok and Υ is a name instantiation, then $P\Upsilon$ ok.
3. If P ok and Υ is a name translation, then $P\Upsilon$ ok.

Proof of Proposition A.4.1 and A.4.2. The proof consists of an induction over the structure of the entities, supporting mathematical inductions over the length of the sequence of substitutions. Again, proofs for Propositions A.4.1 and A.4.2 are presented together. Moreover, Proposition A.3.1 is invoked in the cases where syntactic restriction requirements must be verified (i.e. $(\nu\, a@s)\,\mathrm{X}$, $(\nu\, a)\,\mathrm{X}$ and $u?(\tilde{x})P$). Note that by definition of X ok, the i.h. may be applied to all the subterms of X.

1. Processes

   - $\mathbf{0}$

     - $\mathbf{0}$ $\mathsf{ok} \Rightarrow \mathbf{0}[n/m]$ ok
       $(\mathbf{0})[n/m] \stackrel{\text{def}}{=} \mathbf{0}$

     - $\mathbf{0}$ $\mathsf{ok} \Rightarrow (\mathbf{0})[n_1/m_1] \ldots [n_f/m_f]$ ok, for

       $((\mathbf{0})[n_1/m_1] \ldots [n_{f-1}/m_{f-1}])[n_f/m_f]$
       $=$ (i.h.2) $\quad \mathbf{0}[n_f/m_f]$
       $\stackrel{\text{def}}{=} \mathbf{0}$

     It is clear that $(\mathbf{0})[n_1/m_1] \ldots [n_f/m_f]$ belongs to the language. In $(\mathbf{0})[n_1/m_1] \ldots$ $\ldots [n_f/m_f]$ there are no syntactic restrictions to satisfy.

   - $u!\langle\widetilde{n}\rangle$

     - $u!\langle\widetilde{n}\rangle$ $\mathsf{ok} \Rightarrow (u!\langle\widetilde{n}\rangle)[n/m]$ ok,
       since $(u!\langle\widetilde{n}\rangle)[n/m]$
       $\stackrel{\text{def}}{=}[n/m](u)!\langle[n/m](\widetilde{n})\rangle$

       By hypothesis $\{u\widetilde{n}\} \subset (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, and $[n/m]$ is either a name replacement, or a name instantiation. Therefore, $\{[n/m](u)\} \cup \{[n/m](\widetilde{n})\} \subset (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, which is sufficient to verify that $(u!\langle\widetilde{n}\rangle)[n/m]$ belongs to the language. In $[n/m](u)!\langle[n/m](\widetilde{n})\rangle$ there are no syntactic restrictions to satisfy.

     - $u!\langle\widetilde{n}\rangle$ $\mathsf{ok} \Rightarrow (u!\langle\widetilde{n}\rangle)[n_1/m_1] \ldots [n_f/m_f]$ ok, since

       By the i.b.2, $(u!\langle\widetilde{n}\rangle)[n_1/m_1]$ ok. By definition $(u!\langle\widetilde{n}\rangle)[n_1/m_1]$ is of the form $u'!\langle n'n'\rangle$, therefore by i.h.2, $(u!\langle\widetilde{n}\rangle)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

   - $Q_1 \mid Q_2$

     - $Q_1 \mid Q_2$ $\mathsf{ok} \Rightarrow (Q_1 \mid Q_2)[n/m]$ ok, since

       $(Q_1 \mid Q_2)[n/m]$
       $\stackrel{\text{def}}{=}(Q_1)[n/m] \mid (Q_2)[n/m]$

       By i.h.1 we have that $(Q_1)[n/m]$ ok and $(Q_2)[n/m]$ ok. Therefore $((Q_1)[n/m] \mid$ $(Q_2)[n/m])$ ok.

- $Q_1 \mid Q_2$ $\mathsf{ok} \Rightarrow (Q_1 \mid Q_2)[n_1/m_1]\dots[n_f/m_f]$ ok, since

  By i.b.2, $(Q_1 \,|\, Q_2)[n_1/m_1]$ ok. Since by definition $(Q_1 \,|\, Q_2)[n_1/m_1]$ is of the form $Q_1' \mid Q_2'$, then by i.h.2, $(Q_1 \mid Q_2)[n_1/m_1][n_2/m_2]\dots[n_f/m_f]$ ok.

- $(\nu\, t)\, Q$

  - $(\nu\, t)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, t)\, Q)[n/m]$ ok, for

    * If $t \in m$,

      $((\nu\, t)\, Q)[n/m] \stackrel{\text{def}}{=} (\nu\, t)\, Q$

    * If $t \notin m$, and $t \notin n$ or $m \notin \mathsf{fn}(Q)$

      $((\nu\, t)\, Q)[n/m] \stackrel{\text{def}}{=} (\nu\, t)\, Q[n/m]$

      By i.h.1 we have that $(Q)[n/m]$ ok. Therefore $(\nu\, t)\,(Q)[n/m]$ ok.

    * If $t \notin m$, and $t \in n$ and $m \in \mathsf{fn}(M)$, where s is fresh

      $((\nu\, t)\, Q)[n/m] \stackrel{\text{def}}{=} (\nu\, s)\, Q[s/t][n/m]$

      By i.h.1 we have that $(Q)[s/t][n/m]$ ok. Therefore $(\nu\, t)\,(Q)[s/t][n/m]$ ok.

  - $(\nu\, t)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, t)\, Q)[n_1/m_1]\dots[n_f/m_f]$ ok, since

    By i.b.2, $((\nu\, t)\, Q)[n_1/m_1]$ ok. Since by definition $((\nu\, t)\, Q)[n_1/m_1]$ is of the form $(\nu\, t')\, Q'$, then by i.h.2, $((\nu\, t)\, Q)[n_1/m_1][n_2/m_2]\dots[n_f/m_f]$ ok.

- $(\nu\, a)\, Q$

  - $(\nu\, a)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, a)\, Q)[n/m]$ ok, for

    * If $m = a$,

      $((\nu\, a)\, Q)[n/a] \stackrel{\text{def}}{=} (\nu\, a)\, Q$

    * If $m \neq a$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$,

$((\nu\, a)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, a)\, Q[n/m]$

We must verify that if $m \neq a$, and ($a \notin n$ or $m \notin \mathsf{fn}(Q)$) then $(\nu\, a)\, Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok.
Therefore $(\nu\, a)\, (Q)[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1b), i.e.:
$\forall t : a@t \notin \mathsf{fn}((Q)[n/m])$

$\mathsf{fn}((Q)[n/m])$
$=$ (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[n/m]$, because $[n/m]$ is either a name replacement or a name instantiation
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\})$

Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. Now it suffices to note that: $\forall t : (n \neq a@t\|$ $\|m \notin \mathsf{fn}(Q))$ and $a@t \notin \mathsf{fn}(Q) \Rightarrow a@t \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

$*$ If $m \neq a$, and $a \in n$ and $m \in \mathsf{fn}(Q)$ where b is fresh,

$((\nu\, a)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, b)\, Q[b/a][n/m]$

We must verify that if $m \neq a$, and ($a \notin n$ or $m \notin \mathsf{fn}(Q)$) then $(\nu\, a)\, Q[n/m]$ ok. By i.h.1 we have that $(Q)[b/a][n/m]$ ok. Therefore
$(\nu\, a)\, (Q)[b/a][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1b),i.e.:
$\forall t : a@t \notin \mathsf{fn}((Q)[b/a][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

$\mathsf{fn}((Q)[b/a][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
$=$ (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[b/a][n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, since $[n/m]$ is either a name replacement or name instantiation and $[b/a]$ is a name replacement
$\overset{\text{def}}{=}\{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})\}$
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)\}\}$
$= \{m_1[b/a][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. Now it suffices to note that: $\forall t : (n \neq a@t\|m \notin \mathsf{fn}(Q))$ and $a@t \notin \mathsf{fn}(Q) \Rightarrow a@t \notin \{m_1[b/a][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

– $(\nu\, a)\, Q$ ok $\Rightarrow ((\nu\, a)\, Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, for

By i.b.2, $((\nu\, a)\, Q)[n_1/m_1]$ ok. Since by definition $((\nu\, a)\, Q)[n_1/m_1]$ is of the form $(\nu\, a')\, Q'$, then by i.h.2 $((\nu\, a)\, Q)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

- $(\nu\, a@t)\, Q$

  - $(\nu\, a@t)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, a@t)\, Q)[n/m]$ ok, for

    * If $m = a@t$,

      $((\nu\, a@t)\, Q)[n/a@t] \overset{\text{def}}{=} (\nu\, a@t)\, Q$

    * If $m = t$,

      $((\nu\, a@t)\, Q)[n/t] \overset{\text{def}}{=} (\nu\, a@n)\, Q[n/t]$

      We must verify that if $(n \in \mathcal{S})$ then $(\nu\, a@n)\, Q[n/m]$ ok. By i.h.1 we have that $(Q)[n/m]$ ok. Therefore, $(\nu\, a@n)\, (Q)[n/m]$ ok iff it satisfies syntactic restriction (2.5.1a), that is: $a \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

      $\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
      $=$ (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$ since $[n/m]$ is either a name replacement or a name instantiation
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

      We know that P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that: $n \in \mathcal{S}$ and $a \notin \mathsf{fn}(Q) \Rightarrow a \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

    * $If m \neq a@t$ and $m \neq t$, and $n \notin \{a, a@t\}$ or $m \notin \mathsf{fn}(Q)$
      $((\nu\, a@t)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, a@t)\, Q[n/m]$

      We must verify that, if $m \neq a@t$ and $m \neq t$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a@t)\, Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok. Thus, $(\nu\, a@t)\, (Q)[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1a), that is: $a \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

      $\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
      $=$ (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$ since $[n/m]$ is either a name replacement or a name instantiation
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

      We know that P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices now to note that: $(n \neq a$ or $m \notin \mathsf{fn}(Q))$ and $a \notin \mathsf{fn}(Q) \Rightarrow a \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

    * If $m \neq a@t$ and $m \neq t$, and $n \in \{a, a@t\}$ and $m \in \mathsf{fn}(M)$ where b is fresh,

$((\nu\, a@t)\, M)[n/m] \overset{\text{def}}{=} (\nu\, b@t)\, M[b@t/a@t][n/m]$

We must check that, if $m \neq a@t$ and $m \neq t$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a@t)\, Q[b@t/a@t][n/m]$ ok. By i.h. we have $(Q)[b@t/a@t][n/m]$ ok. Thus $(\nu\, a@t)\, (Q)[b@t/a@t][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1a), that is: $a \notin \mathsf{fn}((Q)[b@t/a@t][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

$\mathsf{fn}((Q)[b@t/a@t][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
$=$ (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[b@t/a@t][n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, since $[n/m]$ is either a name replacement or a name instantiation and $[b@t/a@t]$ is a name replacement
$\overset{\text{def}}{=} \{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\}[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
$\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\}\}$

We know that P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that: ($n \neq a$ or $m \notin \mathsf{fn}(Q)$) and $a \notin \mathsf{fn}(Q) \Rightarrow a \notin \{m_1[b@t/a@t][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

– $(\nu\, a@t)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, a@t)\, Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, since

By i.b.2 $((\nu\, a@t)\, Q)[n_1/m_1]$ ok. Since, by definition $((\nu\, a@t)\, Q)[n_1/m_1]$ is of the form $(\nu\, a'@t)\, Q'$ then, by i.h.2 $((\nu\, a@t)\, Q)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

- $u?(\tilde{a})Q$

  – $u?(\tilde{a})Q$ $\mathsf{ok} \Rightarrow (u?(\tilde{a})Q)[n/m]$ ok, since

  ∗ If $a_i \in \{\tilde{a}\}$,

  $(u?(\tilde{a})Q)[n/a_i] \overset{\text{def}}{=} u[n/a_i]?(\tilde{a})Q$

  ∗ If $m \notin \{\tilde{a}\}$, and $\forall i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$

  $(u?(\tilde{a})Q)[n/m] \overset{\text{def}}{=} u[n/m]?(\tilde{a})Q[n/m]$

  We must check that, if $n \notin \{\tilde{a}\}$ and $\forall i : a_i \notin n$
  or $m \notin \mathsf{fn}(Q)$ then $u[n/m]?(\tilde{a})Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok. Thus $u[n/m]?(\tilde{a})Q[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1c), that is: $\forall t \forall a \in \{\tilde{a}\} : a@t \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

  $\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

= (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$ since $[n/m]$ is either a name replacement or a name instantiation
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

We know that P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It now suffices to note that: $\forall t \forall a_i \in \{\tilde{a}\} : (n \neq a_i@t$ or $(m \notin \mathsf{fn}(Q))$ and $(a_i@t \notin \mathsf{fn}(Q) \Rightarrow a_i@t \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

$*$ If $m \notin \{\tilde{a}\}$, and for some $i : a_i \in n$ and $m \in \mathsf{fn}(Q)$ where b is fresh

$(u?(\tilde{a})Q)[n/m] \overset{\text{def}}{=} u[n/m]?(a_1 \ldots b \ldots a_n)Q[b/a_i][n/m]$

We must verify that, if $n \notin \{\tilde{a}\}$ and $\forall i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$ then $u[n/m]?(\tilde{a})Q[b/a_i][n/m]$ ok. By i.h. we have $(Q)[b/a_i][n/m]$ ok. Thus $u[n/m]?(\tilde{a})Q[b/a_i][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1c), that is: $\forall t \forall a \in \{\tilde{a}\} : a@t \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

$\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
= (Proposition $A.3.1, A.3.2$) $\mathsf{fn}(Q)[b/a_i][n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$ since $[n/m]$ is either a name replacement or a name instantiation and $[b/a_i]$ is a name replacement
$\overset{\text{def}}{=}\{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\}[n/m]$
$\overset{\text{def}}{=}\{m_1[n/m] \mid m_1 \in \{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\}\}$

Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that: $\forall t \forall a_i \in \{\tilde{a}\} : (n \neq a_i@t$ or $(m \notin \mathsf{fn}(Q))$ and $a_i@t \notin \mathsf{fn}(Q) \Rightarrow a_i@t \notin \{m_1[b/a_i][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

– $u?(\tilde{a})Q$ $\mathsf{ok} \Rightarrow (u?(\tilde{a})Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, since

By i.b.2 $(u?(\tilde{a})Q)[n_1/m_1]$ ok. Since, by definition $(u?(\tilde{a})Q)[n_1/m_1]$ is of the form $u'?(\tilde{a'})Q'$ then, by i.h.2 $(u?(\tilde{a})Q)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

2. Networks

- $\mathbf{0}$, $M_1 \mid M_2$, $(\nu\, t)\, M$, $(\nu\, a@t)\, M$

  This is similar to the case for the processes.

- $s[P]$

- $s[P]$ ok $\Rightarrow (s[P])[n/m]$ ok, since

  * If $m = a@s$,

    $(s[P])[b@s/a@s] \stackrel{\text{def}}{=} s[P[b/a][b@s/a@s]]$

    By i.h.1 we have $(P)[b/a][n/m]$ ok. Thus, $s[P[b/a][n/m]]$ ok.

  * If $m = s$,

    $(s[P])[n/s] \stackrel{\text{def}}{=} n[P[n/m]]$

    By i.h.1 we have $(P)[n/m]$ ok. Thus $n[P[n/m]]$ ok.

  * If $s \notin m$,

    $(s[P])[n/m] \stackrel{\text{def}}{=} s[P[n/m]]$, if $s \notin m$

    By i.h.1 we have $(P)[n/m]$ ok. Thus $s[P[n/m]]$ ok.

- $s[P]$ ok $\Rightarrow s[P]$ ok, since

  By i.b.2 $(s[P])[n_1/m_1]$ ok. Since, by definition $(s[P])[n_1/m_1]$ is of the form $s'[P']$ then, by i.h.2 $(s[P])[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

□

Proof of Proposition A.4.3. This proof is analogous to the previous one. It differs from that of Proposition A.4.1 only in the invocation of Proposition A.3.1 (in this case Proposition A.3.3). Note that by definition of X ok, the i.h. may be applied to all the subterms of X.

1. Processes

   • $(\nu\, a)\, Q$

     - $(\nu\, a)\, Q$ ok $\Rightarrow ((\nu\, a)\, Q)[n/m]$ ok, for

       * If $m = a$,

         $((\nu\, a)\, Q)[n/a] \stackrel{\text{def}}{=} (\nu\, a)\, Q$

* If $m \neq a$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$

  $((\nu\, a)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, a)\, Q[n/m]$

  We must verify that if $m \neq a$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a)\, Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok. Therefore $(\nu\, a)\, (Q)[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1b), i.e.: $\forall t : a@t \notin \mathsf{fn}((Q)[n/m])$

  $\mathsf{fn}((Q)[n/m]) \subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[n/m]$, for m is translatable by n,
  $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \cup \mathsf{sites}(\{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\})$

  Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
  $\forall t : (n \neq a@t \parallel m \notin \mathsf{fn}(Q))$ and $a@t \notin \mathsf{fn}(Q)$
  $\Rightarrow a@t \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
  $\Rightarrow a@t \notin \mathsf{fn}((Q)[n/m])$

* If $m \neq a$, and $a \in n$ and $m \in \mathsf{fn}(Q)$ where b is fresh.

  $((\nu\, a)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, b)\, Q[b/a][n/m]$

  We must verify that if $m \neq a$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a)\, Q[n/m]$ ok. By i.h.1 we have that $(Q)[b/a][n/m]$ ok. Therefore $(\nu\, a)\, (Q)[b/a][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1b), i.e.:
  $\forall t : a@t \notin \mathsf{fn}((Q)[b/a][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

  $\mathsf{fn}((Q)[b/a][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S}) \subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[b/a][n/m] \cap$ $\cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because $[n/m]$ is a name translation, and $[b/a]$ is a name replacement.
  $\overset{\text{def}}{=} \{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)\}[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
  $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \{m_1[b/a] \mid m_1 \in \mathsf{fn}(Q)\}\}$
  $= \{m_1[b/a][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

  Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
  $\forall t : (n \neq a@t \parallel m \notin \mathsf{fn}(Q))$ and $a@t \notin \mathsf{fn}(Q)$
  $\Rightarrow a@t \notin \{m_1[b/a][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
  $\Rightarrow a@t \notin \mathsf{fn}((Q)[b/a][n/m])$

– $(\nu\, a)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, a)\, Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, for

By i.b.2, $((\nu\, a)\, Q)[n_1/m_1]$ ok. Since by definition $((\nu\, a)\, Q)[n_1/m_1]$ is of the same form as $(\nu\, a')\, Q'$, then by i.h.2 $((\nu\, a)\, Q)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

- $(\nu\, a@t)\, Q$

  - $(\nu\, a@t)\, Q$ $\mathsf{ok} \Rightarrow ((\nu\, a@t)\, Q)[n/m]$ ok, for

    * If $m = a@t$,

      $((\nu\, a@t)\, Q)[n/a@t] \overset{\text{def}}{=} (\nu\, a@t)\, Q$

    * If $m = t$,

      $((\nu\, a@t)\, Q)[n/t] \overset{\text{def}}{=} (\nu\, a@n)\, Q[n/t]$

      We must verify that if $n \in \mathcal{S}$ then $(\nu\, a@n)\, Q[n/m]$ ok. By i.h.1 we have that $(Q)[n/m]$ ok. Therefore $(\nu\, a@n)\, (Q)[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1a), i.e.: $a \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

      $\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S}) \subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because m is translatable by n
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

      Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
      $n \in \mathcal{S}$ and $a \notin \mathsf{fn}(Q)$
      $\Rightarrow a \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
      $\Rightarrow a \notin \mathsf{fn}((Q)[n/m])$

    * If $m \neq a@t$ and $m \neq t$, and $n \notin \{a, a@t\}$ or $m \notin \mathsf{fn}(Q)$

      $((\nu\, a@t)\, Q)[n/m] \overset{\text{def}}{=} (\nu\, a@t)\, Q[n/m]$

      We must verify that if $m \neq a@t$ and $m \neq t$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a@t)\, Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok. Therefore $(\nu\, a@t)\, (Q)[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1a), i.e.: $a \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

      $\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
      $\subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because m is translatable by n
      $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

      Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:

$(n \neq a$ or $m \notin \mathsf{fn}(Q))$ and $a \notin \mathsf{fn}(Q)$
$\Rightarrow a \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\} \Rightarrow a \notin \mathsf{fn}((Q)[n/m])$

- If $m \neq a@t$ and $m \neq t$, and n and $m \in \mathsf{fn}(M)$ where b is fresh.

  $((\nu\, a@t)\, M)[n/m] \overset{\text{def}}{=} (\nu\, b@t)\, M[b@t/a@t][n/m]$
  We must verify that if $m \neq a@t$ and $m \neq t$, and $a \notin n$ or $m \notin \mathsf{fn}(Q)$ then $(\nu\, a@t)\, Q[b@t/a@t][n/m]$ ok. By i.h. we have that $(Q)[b@t/a@t][n/m]$ ok. Therefore $(\nu\, a@t)\, (Q)[b@t/a@t][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1a), i.e.: $a \notin \mathsf{fn}((Q)[b@t/a@t][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

  $\mathsf{fn}((Q)[b@t/a@t][n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
  $\subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[b@t/a@t][n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because $[n/m]$ is a name translation, and $[b@t/a@t]$ is a name replacement
  $\overset{\text{def}}{=} \{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\}[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
  $\overset{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \{m_1[b@t/a@t] \mid m_1 \in \mathsf{fn}(Q)\}\}$

  Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
  $(n \neq a$ or $m \notin \mathsf{fn}(Q))$ and $a \notin \mathsf{fn}(Q)$
  $\Rightarrow a \notin \{m_1[b@t/a@t][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
  $\Rightarrow a \notin \mathsf{fn}((Q)[b@t/a@t][n/m])$

– $(\nu\, a@t)\, Q$ ok $\Rightarrow ((\nu\, a@t)\, Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, for

  By i.b.2 $((\nu\, a@t)\, Q)[n_1/m_1]$ ok. Since, by definition, $((\nu\, a@t)\, Q)[n_1/m_1]$ is of the same form as $(\nu\, a'@t)\, Q'$, then by i.h.2 $((\nu\, a@t)\, Q)[n_1/m_1][n_2/m_2] \ldots$
  $\ldots [n_f/m_f]$ ok.

- $u?(\tilde{a})Q$

  – $u?(\tilde{a})Q$ ok $\Rightarrow (u?(\tilde{a})Q)[n/m]$ ok, for
    * If $a_i \in \{\tilde{a}\}$,

      $(u?(\tilde{a})Q)[n/a_i] \overset{\text{def}}{=} u[n/a_i]?(\tilde{a})Q$

    * If $m \notin \{\tilde{a}\}$, and $\forall i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$

      $(u?(\tilde{a})Q)[n/m] \overset{\text{def}}{=} u[n/m]?(\tilde{a})Q[n/m]$

      We must verify that if $n \notin \{\tilde{a}\}$ and $\forall i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$ then $u[n/m]?(\tilde{a})Q[n/m]$ ok. By i.h. we have that $(Q)[n/m]$ ok. Therefore

$u[n/m]?(\tilde{a})Q[n/m]$ ok iff it satisfies the syntactic restriction (2.5.1c), i.e.: $\forall t \forall a \in \{\tilde{a}\} : a@t \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

$\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
$\subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because m is translatable by n
$\stackrel{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$

Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
$\forall t \forall a_i \in \{\tilde{a}\} : (n \neq a_i@t \parallel m \notin \mathsf{fn}(Q))$ and $a_i@t \notin \mathsf{fn}(Q)$
$\Rightarrow a_i@t \notin \{m_1[n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
$\Rightarrow a_i@t \notin \mathsf{fn}((Q)[n/m])$

$*$ If $m \notin \{\tilde{a}\}$, and $\exists i : a_i \in n$ and $m \in \mathsf{fn}(Q)$ where b is fresh.

$(u?(\tilde{a})Q)[n/m] \stackrel{\text{def}}{=} u[n/m]?(a_1 \ldots b \ldots a_n)Q[b/a_i][n/m]$

We must verify that if $n \notin \{\tilde{a}\}$ and $\forall i : a_i \notin n$ or $m \notin \mathsf{fn}(Q)$ then $u[n/m]?(\tilde{a})Q[b/a_i][n/m]$ ok. By i.h. we have that $(Q)[b/a_i][n/m]$ ok. Therefore $u[n/m]?(\tilde{a})Q[b/a_i][n/m]$ ok iff it satisfies the syntactic restriction (2.5.1c), i.e.: $\forall t \forall a \in \{\tilde{a}\} : a@t \notin \mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$

$\mathsf{fn}((Q)[n/m]) \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$
$\subseteq$ (Proposition A.3.3) $\mathsf{fn}(Q)[b/a_i][n/m] \cap (\mathcal{C} \cup \mathcal{C}@\mathcal{S})$, because m is translatable by n and a_i is replaceable by b
$\stackrel{\text{def}}{=} \{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\}[n/m]$
$\stackrel{\text{def}}{=} \{m_1[n/m] \mid m_1 \in \{m_1[b/a_i] \mid m_1 \in \mathsf{fn}(Q)\}\}$

Since P ok, $a@t \notin \mathsf{fn}(Q), \forall t$. It suffices to note that:
$\forall t \forall a_i \in \{\tilde{a}\} : (n \neq a_i@t \parallel m \notin \mathsf{fn}(Q))$ and $a_i@t \notin \mathsf{fn}(Q)$
$\Rightarrow a_i@t \notin \{m_1[b/a_i][n/m] \mid m_1 \in \mathsf{fn}(Q)\}$
$\Rightarrow a_i@t \notin \mathsf{fn}((Q)[n/m])$

– $u?(\tilde{a})Q$ $\mathsf{ok} \Rightarrow (u?(\tilde{a})Q)[n_1/m_1] \ldots [n_f/m_f]$ ok, for

By i.b.2 $(u?(\tilde{a})Q)[n_1/m_1]$ ok. Since by definition $(u?(\tilde{a})Q)[n_1/m_1]$ is of the same form as $u'?(\tilde{a'})Q'$, then by i.h.2 $(u?(\tilde{a})Q)[n_1/m_1][n_2/m_2] \ldots [n_f/m_f]$ ok.

2. Networks

- $(\nu\, a@t)\, M$

Analogous to the respective Process case.

- $s[P]$
  - $s[P]$ ok $\Rightarrow (s[P])[n/m]$ ok, for
    - If $m = a@s$,

      $(s[P])[b@s/a@s] \overset{\text{def}}{=} s[P[b/a][b@s/a@s]]$

      By i.h.1 we have that $(P)[b/a][n/m]$ ok. Therefore $s[P[b/a][n/m]]$ ok.
    - If $m = s$,

      $(s[P])[n/s] \overset{\text{def}}{=} n[P[n/m]]$

      By i.h.1 we have that $(P)[n/m]$ ok. Therefore $n[P[n/m]]$ ok.
    - If $s \notin m$,

      $(s[P])[n/m] \overset{\text{def}}{=} s[P[n/m]]$

      By i.h.1 we have that $(P)[n/m]$ ok. Therefore $s[P[n/m]]$ ok.
  - $s[P]$ ok $\Rightarrow s[P]$ ok, for

    By i.b.2 $(s[P])[n_1/m_1]$ ok. Since by definition $(s[P])[n_1/m_1]$ is of the same form as $s'[P']$, then by i.h.2 $(s[P])[n_1/m_1][n_2/m_2]\ldots[n_f/m_f]$ ok.

□

Proposition A.5 (Alpha congruence preserves the free names and the syntactic restrictions). Let X and Y be both either networks or processes.

1. If X ok and $X \equiv_\alpha Y$, then $\mathsf{fn}(X) = \mathsf{fn}(Y)$.
2. If X ok and $X \equiv_\alpha Y$, then Y ok.

Proof of Proposition A.5.1. This proof uses the name replacement version of Proposition A.3.1 for a verification on each case of *c.o.b.n.*Note that all the substitutions involved in the definition of *c.o.b.n.* are name replacements.

1. Processes

   If $P \equiv_\alpha Q$, then by definition Q is obtained from P by a sequence of *c.o.b.n.*It is thus enough to prove that if Q is obtained from P by a *c.o.b.n.*, then $\mathsf{fn}(P) = \mathsf{fn}(Q)$. Suppose that P_1 and Q_1 are the sub-terms of P and Q, respectively, that are altered by the *c.o.b.n.*P_1 and Q_1 share the same context and therefore, if $\mathsf{fn}(P_1) = \mathsf{fn}(Q_1)$, then $\mathsf{fn}(P) = \mathsf{fn}(Q)$. P_1 and Q_1 may have the following forms:

   - $P_1 = (\nu\, b)\, R$ and $Q_1 = (\nu\, c)\, R[c/b]$ with b(and c) $\in \mathcal{C}$) and $c, c@\_ \notin \mathsf{fn}(R)$.

     – $\mathsf{fn}((\nu\, b)\, R) \overset{\text{def}}{=} \mathsf{fn}(R) \setminus \{b\}$

     – $\mathsf{fn}((\nu\, c)\, R[c/b])$
     $\overset{\text{def}}{=} \mathsf{fn}(R[c/b]) \setminus \{c\}$
     $=$ (Proposition A.3.1) $(\mathsf{fn}(R)[c/b]) \setminus \{c\}$ (because P $\mathsf{ok} \Rightarrow R$ ok, $[c/b]$ is a name replacement
     $\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[c/b] \mid m_1 \in \mathsf{fn}(R)\} \setminus \{c\}$

     Since $b \in \mathcal{C}$,

     $$\overset{\text{def}}{=} \begin{cases} (\mathsf{fn}(R) \setminus \{b\} \cup \{c\}) \setminus \{c\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) \setminus \{c\} & \text{otherwise} \end{cases}$$

     By hypothesis, $c \notin \mathsf{fn}(R)$, and therefore

     $$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) & \text{otherwise} \end{cases}$$

     $= \mathsf{fn}(R) \setminus \{b\}$

   - $P_1 = (\nu\, r_1)\, R$ and $Q_1 = (\nu\, t_1)\, R[t_1/r_1]$ with $r_1, t_1 \in \mathcal{S}$ and $t_1 \notin \mathsf{fn}(R)$.

     – $\mathsf{fn}((\nu\, r_1)\, R) \overset{\text{def}}{=} \mathsf{fn}(R) \setminus \{\_@r_1, r_1\}$

     – $\mathsf{fn}((\nu\, t_1)\, R[t_1/r_1])$
     $\overset{\text{def}}{=} \mathsf{fn}(R[t_1/r_1]) \setminus \{\_@t_1, t_1\}$
     $=$ (Proposition A.3.1) $(\mathsf{fn}(R)[t_1/r_1]) \setminus \{\_@t_1, t_1\}$ (since P $\mathsf{ok} \Rightarrow R$ ok, $[t_1/r_1]$ is a name replacement)
     $\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[t_1/r_1] \mid m_1 \in \mathsf{fn}(R)\} \setminus \{\_@t_1, t_1\}$

     Since $r_1 \in \mathcal{S}$,

$= \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 \in \mathcal{C}@\mathcal{S}\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1[t_1/r_1] \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@r_1\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@t, t \neq r_1\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1[t_1/r_1] \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = r_1\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = t, t \neq r_1\} \setminus \{\_@t_1, t_1\}$

Since $r_1 \in \mathcal{S}$,

$= \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 \in \mathcal{C}\} \cup$
$\cup \{a@t_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@r_1\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@t, t \neq r_1\} \cup$
$\cup \{t_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = r_1\} \setminus \{\_@t_1, t_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = t, t \neq r_1\}$

Since $r_1 \in \mathcal{S}$ and $r_1 \notin \mathsf{fn}(R)$ and using Lemma A.2.3

$= \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 \in \mathcal{C}@\mathcal{S}\} \setminus \{\_@r_1, r_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@r_1\} \setminus \{\_@r_1, r_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = a@t, t \neq r_1\} \setminus \{\_@r_1, r_1\} \cup$
$\cup \{t_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = r_1\} \setminus \{\_@r_1, r_1\} \cup$
$\cup \{m_1 \mid m_1 \in \mathsf{fn}(R) \text{ and } m_1 = t, t \neq r_1\} \setminus \{\_@r_1, r_1\} \cup$
$= \mathsf{fn}(R) \setminus \{\_@r_1, r_1\}$

- $P_1 = (\nu\, b@r)\, R$ and $Q_1 = (\nu\, c@r)\, R[c@r/b@r]$ and $c, c@r \notin \mathsf{fn}(R)$.

  – $\mathsf{fn}((\nu\, b@r)\, R) \overset{\text{def}}{=} \mathsf{fn}(R) \setminus \{b@r\} \cup \{r\}$

  – $\mathsf{fn}((\nu\, c@r)\, R[c@r/b@r])$
  $\overset{\text{def}}{=} \mathsf{fn}(R[c@r/b@r]) \setminus \{c@r\} \cup \{r\}$
  $=$ (Proposition $A.3.1$) $(\mathsf{fn}(R)[c@r/b@r]) \setminus \{c@r\} \cup \{r\}$ (because P $\mathsf{ok} \Rightarrow R$ ok, and $[c@r/b@r]$ is a name replacement)
  $\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[c@r/b@r] \mid m_1 \in \mathsf{fn}(R)\} \setminus \{c@r\} \cup \{r\}$

  Since $b@r \in \mathcal{C}@\mathcal{S}$,

$$\overset{\text{def}}{=} \begin{cases} (\mathsf{fn}(R) \setminus \{b@r\} \cup \{c@r\}) \setminus \{c@r\} \cup \{r\} & \text{if } b@r \in \mathsf{fn}(R) \\ \mathsf{fn}(R) \setminus \{c@r\} \cup \{r\} & \text{otherwise} \end{cases}$$

  By hypothesis, $c@r \notin \mathsf{fn}(R)$, then

$$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b@r\} \cup \{r\} & \text{if } b@r \in \mathsf{fn}(R) \\ \mathsf{fn}(R) \cup \{r\} & \text{otherwise} \end{cases}$$

$= \mathsf{fn}(R) \setminus \{b@r\} \cup \{r\}$

- $P_1 = (u?(b_1 \ldots b \ldots b_n)R)$ and $Q_1 = (u?(b_1 \ldots c \ldots b_n)R[c/b])$
  $c, c@\_ \notin \mathsf{fn}(R)$
  $c \notin \{b_1 \ldots b_n\}$.

  – $\mathsf{fn}(u?(b_1 \ldots b \ldots b_n)R) \overset{\text{def}}{=} \{u\} \cup \mathsf{fn}(R) \setminus \{b_1 \ldots b \ldots b_n\}$

  – $\mathsf{fn}(u?(b_1 \ldots c \ldots b_n)R\{c/b\})$
  $\overset{\text{def}}{=} \{u\} \cup \mathsf{fn}(R[c/b]) \setminus \{b_1 \ldots c \ldots b_n\}$
  $=$ (Proposition $A.3.1$) $\{u\} \cup (\mathsf{fn}(R)[c/b]) \setminus \{b_1 \ldots c \ldots b_n\}$ (since P $\mathsf{ok} \Rightarrow R$ ok, and $[c/b]$ is a name replacement)
  $\overset{\text{def}}{=} \{u\} \cup \{m_1[c/b] \mid m_1 \in \mathsf{fn}(R)\} \setminus \{b_1 \ldots c \ldots b_n\}$

  Since $b \in \mathcal{C}$, and P ok and $\{b_1 \ldots c \ldots b_n\}$ pairwise distinct,

  $$\overset{\text{def}}{=} \begin{cases} (\mathsf{fn}(R) \setminus \{b\} \cup \{c\}) \setminus \{b_1 \ldots c \ldots b_n\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) \setminus \{b_1 \ldots c \ldots b_n\} & \text{otherwise} \end{cases}$$

  by hypothesis, $c \notin \mathsf{fn}(R)$, thus

  $$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b_1 \ldots b \ldots b_n\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) \setminus \{b_1 \ldots b_n\} & \text{otherwise} \end{cases}$$

  $= \mathsf{fn}(R) \setminus \{b_1 \ldots b \ldots b_n\}$

2. Networks

If $M \equiv_\alpha N$, then by definition N is obtained from M by a sequence of *c.o.b.n.*It is enough to show that, if N is obtained from M by a *c.o.b.n.*, then $\mathsf{fn}(M) = \mathsf{fn}(N)$. Assume that M_1 and N_1 are the sub-terms of M and N, respectively, that are altered by the *c.o.b.n.*. M_1 and N_1 share the same context, thus $\mathsf{fn}(M_1) = \mathsf{fn}(N_1)$, then $\mathsf{fn}(M) = \mathsf{fn}(N)$. M_1 and N_1 may have the following form:

- $M_1 = (\nu\, g)\, R$ and $N_1 = (\nu\, h)\, R[h/g]$, $[h/g]$ is a name replacement and $h \notin \mathsf{fn}(R)$.

The proof is similar to the second and third cases for processes.

□

Proof of Proposition A.5.2. Analogously to the previous one, this proof uses the name replacement version of Proposition A.3.1 for a verification on each case of *c.o.b.n.*Furthermore, it uses the name replacement version of Proposition A.4.2.

1. Processes

If $P \equiv_\alpha Q$, then by definition Q is obtained from P by a sequence of *c.o.b.n.*It is then enough to show that if Q is obtained from P by a *c.o.b.n.*, and P ok, then Q ok. Assume that P_1 and Q_1 are the sub-terms of P and Q, respectively, that are altered by the $c.o.b.n.P_1$ and Q_1 share the same context and thus we have P_1 ok $\Rightarrow Q_1$ ok, and P ok $\Rightarrow Q$ ok. P_1 and Q_1 may have the following form:

- $P_1 = (\nu\, b)\, R$ and $Q_1 = (\nu\, c)\, R[c/b]$ with $b(and c) \in \mathcal{C}@\mathcal{S})$, and c,$c@_- \notin \mathsf{fn}(R)$.

  By hypothesis P ok, then R ok. Using Proposition A.4.1, if R ok then $R[c/b]$ ok. It is enough to check that Q_1 satisfies the restriction (2.5.1b), that is, $\forall t : c@t \notin \mathsf{fn}(R[c/b])$.

  $\mathsf{fn}(R[c/b])$
  $=$ (Proposition A.3.1) $(\mathsf{fn}(R)[c/b])$
  $\overset{\text{def}}{=}$(Lemma A.2.3) $\{m_1[c/b] \mid m_1 \in \mathsf{fn}(R)\}$

  Since $b \in \mathcal{C}$,
  $$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b\} \cup \{c\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) & \text{otherwise} \end{cases}$$
  By hypothesis, $\forall t : c@t \notin \mathsf{fn}(R)$, then $\forall t : c@t \notin \mathsf{fn}(R) \setminus \{b\} \cup \{c\}$.

- $P_1 = (\nu\, r)\, R$ and $Q_1 = (\nu\, t)\, R\{t/r\}$ with $r, t \in \mathcal{S}$ and $t \notin \mathsf{fn}(R)$.

  By hypothesis P ok, then R ok. Using Proposition A.4.1, if R ok then $R[t/r]$ ok. There are no more conditions to satisfy.

- $P_1 = (\nu\, b@r)\, R$ and $Q_1 = (\nu\, c@r)\, R[c@r/b@r]$ and $c, c@r \notin \mathsf{fn}(R)$.

  By hypothesis P ok, then R ok. Using Proposition A.4.1, if R ok then $R[c@r/b@r]$ ok. It is enough to check that Q_1 satisfies restriction (2.5.1a), that is, that $c \notin \mathsf{fn}(R[c/b])$.

  $\mathsf{fn}(R[c@r/b@r])$
  $=$ (Proposition A.3.1) $\mathsf{fn}(R)[c@r/b@r]$
  $\overset{\text{def}}{=}\{m_1[c@r/b@r] \mid m_1 \in \mathsf{fn}(R)\}$

  Since $b@r \in \mathcal{C}@\mathcal{S}$
  $$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b@r\} \cup \{c@r\} & \text{if } b@r \in \mathsf{fn}(R) \\ \mathsf{fn}(R) & \text{otherwise} \end{cases}$$

By hypothesis, $c \notin \mathsf{fn}(R)$, then $c \notin \mathsf{fn}(R) \setminus \{b@r\} \cup \{c@r\}$.

- $P_1 = (u?(b_1 \ldots b \ldots b_n)R)$ and $Q_1 = (u?(b_1 \ldots c \ldots b_n)R[c/b])$ and $c, c@\_ \notin \mathsf{fn}(R)$ and $c \notin \{b_1 \ldots b_n\}$.

  By hypothesis P ok, then R ok. Using Proposition A.4.1, if R ok then $R[c/b]$ ok. It is enough to check that Q_1 satisfies restrictions (2.5.1c), that is, that $c \notin \{b_1 \ldots b_n\}$ (true by hypothesis) and that $\forall t : c@t \notin \mathsf{fn}(R[c/b])$.

  $\mathsf{fn}(R[c/b])$
  $=$ (Proposition $A.3.1$)$(\mathsf{fn}(R)[c/b])$
  $\overset{\text{def}}{=}\{m_1[c/b] \mid m_1 \in \mathsf{fn}(R)\}$

  Since $b \in \mathcal{C}$
  $$\overset{\text{def}}{=} \begin{cases} \mathsf{fn}(R) \setminus \{b\} \cup \{c\} & \text{if } b \in \mathsf{fn}(R) \\ \mathsf{fn}(R) & \text{otherwise} \end{cases}$$
  By hypothesis, $\forall t : c@t \notin \mathsf{fn}(R)$, since $\forall t : c@t \notin \mathsf{fn}(R) \setminus \{b\} \cup \{c\}$.

2. Networks

If $M \equiv_\alpha N$, then by definition N is obtained from M by a sequence of *c.o.b.n.*It is enough to show that if N is obtained from M by a *c.o.b.n.*, and M ok, then N ok. Assume that M_1 and N_1 are the sub-terms of M and N, respectively, that are altered by the *c.o.b.n.*M_1 and N_1 share the same context and thus M_1 $\mathsf{ok} \Rightarrow N_1$ ok, then M $\mathsf{ok} \Rightarrow N$ ok. M_1 and N_1 may have the following form:

- $M_1 = (\nu\, r)\, L$ and $N_1 = (\nu\, t)\, L[t/r]$ and $t \notin \mathsf{fn}(L)$.

  By hypothesis M ok, then L ok. Using Proposition A.4.1, if L ok then $L[t/r]$ ok. There are no more conditions to satisfy.

- $M_1 = (\nu\, b@r)\, L$ and $N_1 = (\nu\, c@r)\, L[c@r/b@r]$ and $c, c@r \notin \mathsf{fn}(L)$.

  By hypothesis M ok, then L ok. Using Proposition A.4.1, if L ok then $L[c@r/b@r]$ ok. there are no more conditions to satisfy.

$\square$

Proposition A.6 (Structural congruence preserves the free names and the syntactic restrictions). Let X and Y both be either networks or processes.

1. If X ok and $X \equiv Y$, then $\mathsf{fn}(X) = \mathsf{fn}(Y)$.
2. If X ok and $X \equiv Y$, then Y ok.

Proof of Proposition A.6.1. This proof performs a verification on each rule in the definition of structural congruence. In the process of proving this proposition, the side conditions of the mentioned rules may be understood. Of course, the cases of the ALPHA rules just follow from Proposition A.5.

1. Processes

   [SP-ALPHA]

   $$P \equiv Q \text{ if } P \equiv_\alpha Q$$

   By Proposition A.5.1.

   [SP-RESR1]

   $$(\nu\, g)\, (\nu\, h)\, P \equiv (\nu\, h)\, ((\nu\, g)\, P), \text{ if } g \notin \mathsf{fn}(h) \text{ and } h \notin \mathsf{fn}(g)$$

   - (if $g = t$ and $h = r$)

     The rule may always be applied $t \neq r$.
     $\mathsf{fn}((\nu\, t)\, ((\nu\, r)\, P))$
     $\overset{\text{def}}{=} \mathsf{fn}((\nu\, r)\, P) \setminus \{\_@t, t\}$
     $\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{\_@r, r, \_@t, t\}$

     $\mathsf{fn}((\nu\, r)\, ((\nu\, t)\, P))$
     $\overset{\text{def}}{=} \mathsf{fn}((\nu\, t)\, P) \setminus \{\_@r, r\}$
     $\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{\_@t, t, \_@r, r\}$

   - (if $g = t$ and $h = a@r$)

     This rule may be applied in the first case when $g \notin \mathsf{fn}(h)$, and in the inverse case when $h \notin \mathsf{fn}(g)$, i.e., when, in both cases $t \neq r$.

     $\mathsf{fn}((\nu\, t)\, (\nu\, a@r)\, P)$
     $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a@r)\, P) \setminus \{\_@t, t\}$
     $\overset{\text{def}}{=} (\mathsf{fn}(P) \cup \{r\}) \setminus \{a@r, \_@t, t\}$

     $\mathsf{fn}((\nu\, a@r)\, (\nu\, t)\, P)$
     $\overset{\text{def}}{=} \mathsf{fn}((\nu\, t)\, P) \setminus \{a@r\} \cup \{r\}$

$\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{a@r, \_@t, t\} \cup \{r\}$

The sets are equal whenever $r \neq t$, which is true by hypothesis.

- (if $g = a@s$ and $h = a@r$)

  The rule may be applied as long as $t \neq r$.

  $\mathsf{fn}((\nu\, a@s)\,(\nu\, a@r)\, P)$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a@r)\, P) \setminus \{a@s\}$
  $\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{a@r, a@s\}$

  $\mathsf{fn}((\nu\, a@r)\,(\nu\, a@s)\, P)$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a@s)\, P) \setminus \{a@r\}$
  $\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{a@s, a@r\}$

[SP-RESR2]

$$(\nu\, a)\,((\nu\, s)\, P) \equiv (\nu\, s)\,((\nu\, a)\, P) \text{ if } a@s \notin \mathsf{fn}(P)$$

The rule may always be applied in this case because of the syntactic restriction (definition 2.5.1b) $a@t \notin \mathsf{fn}(P)$, and in the inverse case whenever $a@s \notin \mathsf{fn}(P)$.

$\mathsf{fn}((\nu\, s)\,((\nu\, a)\, P))$
$\overset{\text{def}}{=} \mathsf{fn}((\nu\, a)\, P) \setminus \{\_@s, s\}$
$\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{\_@s, s, a\}$

$\mathsf{fn}((\nu\, a)\,((\nu\, s)\, P))$
$\overset{\text{def}}{=} \mathsf{fn}((\nu\, s)\, P) \setminus \{a\}$
$\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{a, \_@s, s\}$

[SP-RESR3]

$$(\nu\, a)\,((\nu\, u)\, P) \equiv (\nu\, u)\,((\nu\, a)\, P) \text{ if } u \neq a@\_$$

- (if $u = b@s$)

  The rule may always be applied as long as $b \neq a$.

  $\mathsf{fn}((\nu\, a)\,((\nu\, b@s)\, P))$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, b@s)\, P) \setminus \{a\}$
  $\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{b@s, a\}$

$\mathsf{fn}((\nu\, b@s)\,((\nu\, a)\, P))$
$\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a)\, P) \setminus \{b@s\}$
$\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a, b@s\}$

- (if $u = b$)

  As long as the initial process is ok, the rule may always be applied.

  $\mathsf{fn}((\nu\, a)\,((\nu\, b)\, P))$
  $\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, b)\, P) \setminus \{a\}$
  $\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{b, a\}$

  $\mathsf{fn}((\nu\, b)\,((\nu\, a)\, P))$
  $\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a)\, P) \setminus \{b\}$
  $\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a, b\}$

[SP-SCOP1]

$$((\nu\, a)\, P) \mid Q \equiv (\nu\, a)\,(P \mid Q) \text{ if } a, a@\_ \notin \mathsf{fn}(M)$$

The rule may be applied whenever $a, a@\_ \notin \mathsf{fn}(Q)$.

$\mathsf{fn}(((\nu\, a)\, P) \mid Q)$
$\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a)\, P) \cup \mathsf{fn}(Q)$
$\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a\} \cup \mathsf{fn}(Q)$

$\mathsf{fn}((\nu\, a)\,(P \mid Q))$
$\stackrel{\text{def}}{=} \mathsf{fn}(P \mid Q) \setminus \{a\}$
$\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a\} \cup \mathsf{fn}(Q)$ since by hypothesis $a \notin \mathsf{fn}(Q)$.

[SP-SCOP2]

$$((\nu\, a@s)\, P) \mid Q \equiv (\nu\, a@s)\,(P \mid Q) \text{ if } a, a@s \notin \mathsf{fn}(Q)$$

The rule may be applied whenever $a, a@s \notin \mathsf{fn}(Q)$.

$\mathsf{fn}(((\nu\, a@s)\, P) \mid Q)$
$\stackrel{\text{def}}{=} \mathsf{fn}((\nu\, a@s)\, P) \cup \mathsf{fn}(Q)$
$\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a@s\} \cup \mathsf{fn}(Q)$

$\mathsf{fn}((\nu\, a)\,(P \mid Q))$
$\stackrel{\text{def}}{=} \mathsf{fn}(P \mid Q) \setminus \{a@s\}$

$\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{a@s\} \cup \mathsf{fn}(Q)$ since by hypothesis $a@s \notin \mathsf{fn}(Q)$.

[SP-SCOP3]

$$((\nu\, s)\, P) \mid Q \equiv (\nu\, s)\, (P \mid Q) \text{ if } s \notin \mathsf{fn}(Q)$$

The rule may be applied whenever $t \notin \mathsf{fn}(Q)$.

$\mathsf{fn}(((\nu\, s)\, P) \mid Q)$
$\overset{\text{def}}{=} \mathsf{fn}((\nu\, s)\, P) \cup \mathsf{fn}(Q)$
$\overset{\text{def}}{=} \mathsf{fn}(P) \setminus \{\_@s, s\} \cup \mathsf{fn}(Q)$

$\mathsf{fn}((\nu\, s)\, (P \mid Q))$
$\overset{\text{def}}{=} \mathsf{fn}(P \mid Q) \setminus \{\_@s, s\}$
$\overset{\text{def}}{=} (\mathsf{fn}(P) \cup \mathsf{fn}(Q)) \setminus \{\_@s, s\}$
$= \mathsf{fn}(P) \setminus \{\_@s, s\} \cup \mathsf{fn}(Q)$ since by hypothesis $s \notin \mathsf{fn}(Q)$ and by Lemma A.2.2 if $s \notin \mathsf{fn}(Q)$ then $\_@s \notin \mathsf{fn}(Q)$.

2. Networks

By i.h., $(\nu\, n)\, ((\nu\, m)\, N)$ ok, therefore N ok. The cases which remain to be verified are:

[SN-ALPHA]

$$N \equiv M \text{ if } N \equiv_\alpha M$$

By Proposition A.5.1.

[SN-MIGI]

$$s[a@s?(\tilde{b})P] \equiv s[a?(\tilde{b})P]$$

$\mathsf{fn}(s[a@s?(\tilde{b})P])$
$\overset{\text{def}}{=} \mathsf{fn}(a@s?(\tilde{b})P)@s$
$\overset{\text{def}}{=} \{a@r \mid a@r \in \mathsf{fn}(a@s?(\tilde{b})P)\} \cup \{a@s \mid a \in \mathsf{fn}(a@s?(\tilde{b})P)\}$
$\overset{\text{def}}{=} \{a@r \mid a@r \in (\{a@s\} \cup \mathsf{fn}(P) \setminus \{\tilde{b}\})\} \cup \{a@s \mid a \in (\{a@s\} \cup \mathsf{fn}(P) \setminus \{\tilde{b}\})\}$
$\overset{\text{def}}{=} \{a@s\} \cup \{a@r \mid a@r \in \mathsf{fn}(P) \setminus \{\tilde{b}\}\} \cup \{a@s \mid a \in \mathsf{fn}(P) \setminus \{\tilde{b}\}\}$
$= \{a@s\} \cup \mathsf{fn}(P)@s$

$\mathsf{fn}(s[a?(\tilde{b})P])$
$\overset{\text{def}}{=} \mathsf{fn}(a?(\tilde{b})P)@s$
$\overset{\text{def}}{=} \{a@r \mid a@r \in \mathsf{fn}(a?(\tilde{b})P)\} \cup \{a@s \mid a \in \mathsf{fn}(a?(\tilde{b})P)\}$

$\overset{\text{def}}{=}\{a@r \mid a@r \in (\{a\} \cup \mathsf{fn}(P) \setminus \{\tilde{b}\})\} \cup \{a@s \mid a \in (\{a\} \cup \mathsf{fn}(P) \setminus \{\tilde{b}\})\}$
$\overset{\text{def}}{=}\{a@r \mid a@r \in \mathsf{fn}(P) \setminus \{\tilde{b}\}\} \cup \{a@s\} \cup \{a@s \mid a \in \mathsf{fn}(P) \setminus \{\tilde{b}\}\}$
$= \{a@s\} \cup \mathsf{fn}(P)@s$

[SN-MIGO]

$$s[a@s!\langle\widetilde{n}\rangle] \equiv s[a!\langle\widetilde{n}\rangle]$$

$\mathsf{fn}(s[a@s!\langle\widetilde{n}\rangle])$
$\overset{\text{def}}{=}\mathsf{fn}(a@s!\langle\widetilde{n}\rangle)@s$
$\overset{\text{def}}{=}\{a@r \mid a@r \in \mathsf{fn}(a@s!\langle\widetilde{n}\rangle)\} \cup \{a@s \mid a \in \mathsf{fn}(a@s!\langle\widetilde{n}\rangle)\}$
$\overset{\text{def}}{=}\{a@r \mid a@r \in (\{a@s\} \cup \{\widetilde{n}\})\} \cup \{a@s \mid a \in (\{a@s\} \cup \{\widetilde{n}\})\}$
$\overset{\text{def}}{=}\{a@s\} \cup \{a@r \mid a@r \in \{\widetilde{n}\}\} \cup \{a@s \mid a \in \{\widetilde{n}\}\}$
$= \{a@s\} \cup \{\widetilde{n}\}@s$

$\mathsf{fn}(s[a?(\tilde{b})P])$
$\overset{\text{def}}{=}\mathsf{fn}(a!\langle\widetilde{n}\rangle)@s$
$\overset{\text{def}}{=}\{a@r \mid a@r \in \mathsf{fn}(a!\langle\widetilde{n}\rangle)\} \cup \{a@s \mid a \in \mathsf{fn}(a!\langle\widetilde{n}\rangle)\}$
$\overset{\text{def}}{=}\{a@r \mid a@r \in (\{a\} \cup \{\widetilde{n}\})\} \cup \{a@s \mid a \in (\{a\} \cup \{\widetilde{n}\})\}$
$\overset{\text{def}}{=}\{a@r \mid a@r \in \{\widetilde{n}\}\} \cup \{a@s\} \cup \{a@s \mid a \in \{\widetilde{n}\}\}$
$= \{a@s\} \cup \{\widetilde{n}\}@s$

[SN-RESO]

$$(\nu\, g)\, ((\nu\, h)\, N) \equiv (\nu\, h)\, ((\nu\, g)\, N) \text{ if } g \notin \mathsf{fn}(h) \text{ and } h \notin \mathsf{fn}(g)$$

This case is analogous to the Processes rule [SP-RESR1].

[SN-SCOS1]

$$(\nu\, a@s)\, s[P] \equiv s[(\nu\, a)\, P] \text{ if } a@\_ \notin \mathsf{fn}(P)$$

The rule may be applied whenever $a@\_ \notin \mathsf{fn}(P)$.

$\mathsf{fn}((\nu\, a@s)\, s[P])$
$\overset{\text{def}}{=} \mathsf{fn}(s[P]) \setminus \{a@s\}$
$\overset{\text{def}}{=}(\mathsf{fn}(P)@s) \setminus \{a@s\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P)\} \setminus \{a@s\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{a@s\}$
$= \{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{a@s\}\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{a@s\}$
$= (\text{hyp})\{b@r \mid b@r \in \mathsf{fn}(P)\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{a@s\}$

$\mathsf{fn}(s[(\nu\, a)\, P])$
$\overset{\text{def}}{=}\mathsf{fn}((\nu\, a)\, P)@s$
$\overset{\text{def}}{=}(\mathsf{fn}(P) \setminus \{a\})@s$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{a\}\} \cup \{b@s \mid b \in \mathsf{fn}(P) \setminus \{a\}\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P)\} \cup \{b@s \mid b \in \mathsf{fn}(P) \setminus \{a\}\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P)\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{a@s\}$

[SN-SCOS2]

$$(\nu\, a@t)\, s[P] \equiv s[(\nu\, a@t)\, P] \text{ if } a \notin \mathsf{fn}(P)$$

The rule may be applied whenever $a \notin \mathsf{fn}(P)$.

$\mathsf{fn}((\nu\, a@t)\, s[P])$
$\overset{\text{def}}{=}\mathsf{fn}(s[P]) \setminus \{a@t\}$
$\overset{\text{def}}{=}(\mathsf{fn}(P)@s) \setminus \{a@t\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P)\} \setminus \{a@t\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{a@t\}$
$= \text{(hyp)}\ \{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{a@t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P)\}$, for $a \notin \mathsf{fn}(P)$

$\mathsf{fn}(s[(\nu\, a@t)\, P])$
$\overset{\text{def}}{=}\mathsf{fn}((\nu\, a@t)\, P)@s$
$\overset{\text{def}}{=}(\mathsf{fn}(P) \setminus \{a@t\})@s$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{a@t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P) \setminus \{a@t\}\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{a@t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P)\}$

[SN-SCOS3]

$$s[(\nu\, t)\, P] \equiv (\nu\, t)\, s[P] \text{ if } s \neq t$$

The rule may be applied whenever $s \neq t$.

$\mathsf{fn}(s[(\nu\, t)\, P])$
$\overset{\text{def}}{=}\mathsf{fn}((\nu\, t)\, P)@s$
$\overset{\text{def}}{=}(\mathsf{fn}(P) \setminus \{\_@t, t\})@s$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{\_@t, t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P) \setminus \{\_@t, t\}\}$
$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{\_@t, t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P)\}$

$\mathsf{fn}((\nu\, t)\, s[P])$
$\overset{\text{def}}{=}\mathsf{fn}(s[P]) \setminus \{\_@t, t\}$
$\overset{\text{def}}{=}(\mathsf{fn}(P)@s) \setminus \{\_@t, t\}$

$\overset{\text{def}}{=}\{b@r \mid b@r \in \mathsf{fn}(P)\} \setminus \{\_@t, t\} \cup \{b@s \mid b \in \mathsf{fn}(P)\} \setminus \{\_@t, t\}$
$=$ (hyp) $\{b@r \mid b@r \in \mathsf{fn}(P) \setminus \{\_@t, t\}\} \cup \{b@s \mid b \in \mathsf{fn}(P)\}$

[SN-SCOP]

$$((\nu\, g)\, N) \parallel M \equiv (\nu\, g)\, (N \parallel M) \text{ if } g \notin \mathsf{fn}(M)$$

- (if $g = s$)

  The rule may be applied whenever $s \notin \mathsf{fn}(N)$.

  $\mathsf{fn}(((\nu\, s)\, N \parallel M))$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, s)\, N) \cup \mathsf{fn}(M)$
  $\overset{\text{def}}{=} \mathsf{fn}(N) \setminus \{\_@s, s\} \cup \mathsf{fn}(M)$

  $\mathsf{fn}((\nu\, s)\, (N \parallel M))$
  $\overset{\text{def}}{=} \mathsf{fn}(N \parallel M) \setminus \{\_@s, s\}$
  $\overset{\text{def}}{=} (\mathsf{fn}(N) \cup \mathsf{fn}(M)) \setminus \{\_@s, s\}$
  $= \mathsf{fn}(N) \setminus \{\_@s, s\} \cup \mathsf{fn}(M)$, because by hypothesis $s \notin \mathsf{fn}(M)$ and by Lemma A.2.2 if $s \notin \mathsf{fn}(M)$ then $\_@s \notin \mathsf{fn}(M)$.

- (if $n = a@s$)

  The rule may be applied whenever $a@s \notin \mathsf{fn}(M)$.

  $\mathsf{fn}(((\nu\, a@s)\, N) \parallel M)$
  $\overset{\text{def}}{=} \mathsf{fn}((\nu\, a@s)\, N) \cup \mathsf{fn}(M)$
  $\overset{\text{def}}{=} \mathsf{fn}(N) \setminus \{a@s\} \cup \mathsf{fn}(M)$

  $\mathsf{fn}((\nu\, a)\, (N \parallel M))$
  $\overset{\text{def}}{=} \mathsf{fn}(N \parallel M) \setminus \{a@s\}$
  $\overset{\text{def}}{=} \mathsf{fn}(N) \setminus \{a@s\} \cup \mathsf{fn}(M)$ because by hypothesis $a@s \notin \mathsf{fn}(M)$.

$\square$

Proof of Proposition A.6.2. As in the previous proof, this proof also performs a verification on each rule of structural congruence. In the process of proving this proposition, the side conditions of the mentioned rules may be understood. Of course, the cases of the ALPHA rules just follow from Proposition A.5.

1. Processes

[SP-ALPHA]

$$P \equiv Q \text{ if } P \equiv_\alpha Q$$

By Proposition A.5.2.

[SP-RESR1]

$$(\nu\, g)\,((\nu\, h)\, P) \equiv (\nu\, h)\,((\nu\, g)\, P), \text{ if } g \notin \mathsf{fn}(h) \text{ and } h \notin \mathsf{fn}(g)$$

By i.h., $(\nu\, g)\,((\nu\, h)\, P)$ ok, therefore P ok.

- (if $g = t$ and $h = r$)

  There are no syntactic restrictions to satisfy. The rule may be applied whenever $t \neq r$.

- (if $g = t$ and $h = a@r$)

  This rule may be applied in the first case whenever $g \notin \mathsf{fn}(h)$, and in the inverse case whenever $h \notin \mathsf{fn}(g)$, i.e. when in both cases $t \neq r$.

  $((\nu\, t)\,(\nu\, a@r)\, P\ \mathsf{ok})$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $(\nu\, a@r)\, P\ \mathsf{ok}$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $a \notin \mathsf{fn}(P)$
  $\Leftrightarrow P\ \mathsf{ok}$ and $a \notin \mathsf{fn}(P) \setminus \{\_@t\}$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $a \notin \mathsf{fn}((\nu\, t)\, P)$
  (def) $\Leftrightarrow ((\nu\, a@r)\,(\nu\, t)\, P)\ \mathsf{ok}$

- (if $g = a@s$ and $h = a@r$)

  The rule may be applied as long as $t \neq r$.

  $((\nu\, a@s)\,(\nu\, a@r)\, P\ \mathsf{ok})$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $(\nu\, a@r)\, P\ \mathsf{ok}$ and $a \notin \mathsf{fn}((\nu\, a@r)\, P)$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $a \notin \mathsf{fn}(P)$ and $a \notin (\mathsf{fn}(P) \setminus \{a@r\} \cup \{r\})$
  $\Leftrightarrow P\ \mathsf{ok}$ and $a \notin \mathsf{fn}(P)$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $(\nu\, a@s)\, P\ \mathsf{ok}$ and $a \notin \mathsf{fn}(P)$
  $\Leftrightarrow P\ \mathsf{ok}$ and $(\nu\, a@s)\, P\ \mathsf{ok}$ and $a \notin (\mathsf{fn}(P) \setminus \{a@s\} \cup \{s\})$
  (def) $\Leftrightarrow P\ \mathsf{ok}$ and $(\nu\, a@s)\, P\ \mathsf{ok}$ and $a \notin \mathsf{fn}((\nu\, a@s)\, P)$
  (def) $\Leftrightarrow ((\nu\, a@r)\,(\nu\, a@s)\, P)\ \mathsf{ok}$

[SP-RESR2]

$$(\nu\, a)\,((\nu\, s)\, P) \equiv (\nu\, s)\,((\nu\, a)\, P) \text{ if } a@s \notin \mathsf{fn}(P)$$

This rule may always be applied in this case, because by syntactic rule (definition 2.5.1b), $a@t \notin \mathsf{fn}(P)$, and in the inverse case it may be applied whenever $a@s \notin \mathsf{fn}(P)$.

$((\nu\, s)\,(\nu\, a)\, P$ $\mathsf{ok})$
(def) $\Rightarrow P$ ok and $(\nu\, a)\, P$ ok
(def) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P)$
($2.5.1b$) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P) \setminus \{\_@s\}$
(def) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}((\nu\, s)\, P)$
(def) $\Rightarrow ((\nu\, a)\,(\nu\, s)\, P)$ ok

Suppose that $a@s \notin \mathsf{fn}(P)$,

$((\nu\, a)\,(\nu\, s)\, P)$ ok
(def) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}((\nu\, s)\, P)$
(def) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P) \setminus \{\_@s\}$
(hyp) $\Rightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P)$
(def) $\Rightarrow P$ ok and $(\nu\, a)\, P$ ok
(def) $\Rightarrow ((\nu\, s)\,(\nu\, a)\, P$ $\mathsf{ok})$

[SP-RESR3]

$$(\nu\, a)\,((\nu\, u)\, P) \equiv (\nu\, u)\,((\nu\, a)\, P) \text{ if } u \neq a@\_$$

- (if $u = b@s$)

  The rule may be applied only if $b \neq a$.

  $((\nu\, a)\,(\nu\, b@s)\, P)$ ok
  (def) $\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}((\nu\, b@s)\, P)$ and $(\nu\, b@s)\, P$ ok
  (def) $\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P) \setminus \{b@s\}$ and $b \notin \mathsf{fn}(P)$
  (hyp) $\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P)$ and $b \notin \mathsf{fn}(P) \setminus \{a\}$
  (def) $\Leftrightarrow P$ ok and $(\nu\, a)\, P$ ok and $b \notin \mathsf{fn}((\nu\, a)\, P)$
  (def) $\Leftrightarrow P$ ok and $(\nu\, a)\, P$ ok and $(\nu\, b@s)\,(\nu\, a)\, P$ ok
  (def) $\Leftrightarrow ((\nu\, b@s)\,(\nu\, a)\, P$ $\mathsf{ok})$

- (if $u = b$)

  In this case the rule may always be applied.

  $((\nu\, a)\,(\nu\, b)\, P)$ ok
  (def) $\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}((\nu\, b)\, P)$ and $(\nu\, b)\, P$ ok
  (def) $\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P) \setminus \{b\}$ and $b@\_ \notin \mathsf{fn}(P)$

$\Leftrightarrow P$ ok and $a@\_ \notin \mathsf{fn}(P)$ and $b@\_ \notin \mathsf{fn}(P) \setminus \{a\}$
(def) $\Leftrightarrow P$ ok and $(\nu\, a)\, P$ ok and $b@\_ \notin \mathsf{fn}((\nu\, a)\, P)$
(def) $\Leftrightarrow P$ ok and $(\nu\, a)\, P$ ok and $(\nu\, b)\, (\nu\, a)\, P$ ok
(def) $\Leftrightarrow ((\nu\, b)\, (\nu\, a)\, P$ $\mathsf{ok})$

[SP-SCOP1]

$$((\nu\, a)\, P) \mid Q \equiv (\nu\, a)\, (P \mid Q) \text{ if } a, a@\_ \notin \mathsf{fn}(Q)$$

The rule may be applied as long as $a, a@\_ \notin \mathsf{fn}(Q)$.

Suppose that $a@t \notin \mathsf{fn}(Q), \forall t$,

$((\nu\, a)\, P \mid Q)$ ok
(def) $\Rightarrow (\nu\, a)\, P$ ok and Q ok and P ok
(def) $\Rightarrow a@\_ \notin \mathsf{fn}(P)$ and Q ok and P ok
(hyp) $\Rightarrow a@\_ \notin \mathsf{fn}(P \mid Q)$ and Q ok and P ok
(def) $\Rightarrow ((\nu\, a)\, P \mid Q)$ ok

$((\nu\, a)\, P \mid Q)$ ok
(def) $\Rightarrow a@\_ \notin \mathsf{fn}(P \mid Q)$ and Q ok and P ok
$\Rightarrow a@\_ \notin \mathsf{fn}(P)$ and Q ok and P ok
(def) $\Rightarrow (\nu\, a)\, P$ ok and Q ok and P ok
(def) $\Rightarrow (\nu\, a)\, P \mid Q$ ok

[SP-SCOP2]

$$((\nu\, a@s)\, P) \mid Q \equiv (\nu\, a@s)\, (P \mid Q) \text{ if } a, a@s \notin \mathsf{fn}(Q)$$

The rule may be applied as long as $a, a@s \notin \mathsf{fn}(Q)$.

Suppose that $a \notin \mathsf{fn}(Q)$,

$((\nu\, a@s)\, P) \mid Q$ ok
(def) $\Rightarrow (\nu\, a@s)\, P$ ok and Q ok and P ok
(def) $\Rightarrow a \notin \mathsf{fn}(P)$ and Q ok and P ok
(hyp) $\Rightarrow a \notin \mathsf{fn}(P \mid Q)$ and Q ok and P ok
(def) $\Rightarrow (\nu\, a@s)\, (P \mid Q)$ ok

$(\nu\, a@s)\, (P \mid Q)$ ok
(def) $\Rightarrow a \notin \mathsf{fn}(P \mid Q)$ and Q ok and P ok
$\Rightarrow a \notin \mathsf{fn}(P)$ and Q ok and P ok
(def) $\Rightarrow (\nu\, a@s)\, P$ ok and Q ok and P ok
(def) $\Rightarrow (\nu\, a@s)\, P \mid Q$ ok

[SP-SCOP3]

$$((\nu\, s)\, P) \mid Q \equiv (\nu\, s)\, (P \mid Q) \text{ if } s \notin \mathsf{fn}(Q)$$

The rule may be applied as long as $t \notin \mathsf{fn}(Q)$. There are no syntactic restrictions to satisfy.

2. Networks

[SN-ALPHA]

$$N \equiv M \text{ if } N \equiv_\alpha M$$

By Proposition A.5.2.

[SN-MIGI]

$$s[a@s?(\tilde{b})P] \equiv s[a?(\tilde{b})P]$$

There are no syntactic restrictions to satisfy.

[SN-MIGO]

$$s[a@s!\langle\widetilde{n}\rangle] \equiv s[a!\langle\widetilde{n}\rangle]$$

There are no syntactic restrictions to satisfy.

[SN-RESO]

$$(\nu\, g)\, ((\nu\, h)\, N) \equiv (\nu\, h)\, ((\nu\, g)\, N) \text{ if } g \notin \mathsf{fn}(h) \text{ and } h \notin \mathsf{fn}(g)$$

By i.h., $(\nu\, n)\, ((\nu\, m)\, N)$ ok, therefore N ok. There are no syntactic restrictions to satisfy.

[SN-SCOS1]

$$(\nu\, a@s)\, s[P] \equiv s[(\nu\, a)\, P] \text{ if } a@\_ \notin \mathsf{fn}(P)$$

The rule may be applied as long as $a@\_ \notin \mathsf{fn}(P)$.

$(s[(\nu\, a)\, P]\ \mathsf{ok})$
(def) $\Rightarrow (\nu\, a)\, P\ \mathsf{ok}$
(def) $\Rightarrow P\ \mathsf{ok}$
(def) $\Rightarrow s[P]\ \mathsf{ok}$
(def) $\Rightarrow (\nu\, a@s)\, s[P]\ \mathsf{ok}$

Suppose that $\forall t : a@t \notin \mathsf{fn}(P)$,

$((\nu\, a@s)\, s[P])$ ok
(def) $\Rightarrow s[P]$ ok
(def) $\Rightarrow P$ ok
(hyp) $\Rightarrow (\nu\, a)\, P$ ok
(def) $\Rightarrow s[(\nu\, a)\, P]$ ok

[SN-SCOS2]

$$(\nu\, a@t)\, s[P] \equiv s[(\nu\, a@t)\, P] \text{ if } a \notin \mathsf{fn}(P)$$

The rule may be applied as long as $a \notin \mathsf{fn}(P)$.

$(s[(\nu\, a@t)\, P]$ ok$)$
(def) $\Rightarrow (\nu\, a@t)\, P$ ok
(def) $\Rightarrow P$ ok
(def) $\Rightarrow s[P]$ ok
(def) $\Rightarrow ((\nu\, a@t)\, s[P])$ ok

Suppose that $\forall t : a@t \notin \mathsf{fn}(P)$

$((\nu\, a@t)\, s[P])$ ok
(def) $\Rightarrow s[P]$ ok
(def) $\Rightarrow P$ ok
(hyp) $\Rightarrow (\nu\, a@t)\, P$ ok
(def) $\Rightarrow s[(\nu\, a@t)\, P]$ ok

[SN-SCOS3]

$$s[(\nu\, t)\, P] \equiv (\nu\, t)\, s[P] \text{ if } s \neq t$$

The rule may always be applied. There are no syntactic restrictions to satisfy.

[SN-SCOP]

$$((\nu\, g)\, N) \parallel M \equiv (\nu\, g)\, (N \parallel M) \text{ if } g \notin \mathsf{fn}(Q)$$

The rule may be applied whenever $g \notin \mathsf{fn}(P)$.

$\square$

Proposition A.7 (Reduction preserves the free names and the syntactic restrictions). Let X and Y both be either networks or processes.

1. If X ok and $X \rightarrow Y$, then $\mathsf{fn}(X) \supseteq \mathsf{fn}(Y)$.

2. If X ok and $X \to Y$, then Y ok.

Proof of Proposition A.7.1. The proof consists of an induction on the derivation of the reduction step. The delicate cases are the axioms of reduction RP-COMM, RN-MIGO and RN-MIGI. Proposition A.3.1 is used.

The induction steps concerning rules STR use Proposition A.6. In the cases of CONT a second induction on the structure of the contexts may be used; the first result is useful for proving this case of the second result.

1. Processes. By induction on the structure of the derivation of $P \to Q$. This result is true for the process axioms:

   [RP-COMM]
   $$P = (a?(\tilde{b})Q \mid a!\langle\tilde{v}\rangle); \quad P \to Q\{\tilde{v}/\tilde{b}\}$$

   Note that communication is defined only if $|\{\tilde{v}\}| = |\{\tilde{b}\}|$.

   - If $\{\tilde{v}\} \cap \{\tilde{b}\} = \emptyset$

     $\mathsf{fn}(P) \overset{\text{def}}{=} \{a\} \cup \mathsf{fn}(Q) \setminus \{b_1 \dots b_n\} \cup \mathsf{fn}(\{v_1 \dots v_n\})$

     $\mathsf{fn}(Q\{v_1/b_1 \dots v_n/b_n\})$
     $\overset{\text{def}}{=} \mathsf{fn}(Q[v_1/b_1] \dots [v_n/b_n])$
     $=$ (Proposition A.3.2) $\mathsf{fn}(Q)[v_1/b_1] \dots [v_n/b_n]$
     $=$ (Lemma A.2.4) $\{m_1[v_1/b_1] \dots [v_n/b_n] \mid m_1 \in \mathsf{fn}(Q)\} \cup$
     $\cup\, \mathsf{sites}(\{m_1[b_1/b_1] \dots [v_n/b_n] \mid m_1 \in \mathsf{fn}(Q)\})$

     Since by hypothesis $\{v_1, \dots, v_n\} \cap \{b_1, \dots, b_n\} = \emptyset$, and since $\{b_1, \dots, b_n\} \subset \mathcal{C}$, the substitutions in the sequence $[v_1/b_1] \dots [v_n/b_n]$ don't interfere with one another. Therefore,

     $\subseteq \mathsf{fn}(Q) \setminus \{b_1, \dots, b_n\} \cup \{v_1, \dots, v_n\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{b_1, \dots, b_n\} \cup \{v_1, \dots, v_n\})$

     Since $\forall v : v \in \mathsf{fn}(v)$

     $\subseteq \mathsf{fn}(Q) \setminus \{b_1, \dots, b_n\} \cup \mathsf{fn}(\{v_1, \dots, v_n\}) \cup \mathsf{sites}(\mathsf{fn}(Q)) \setminus \{b_1, \dots, b_n\} \cup \{v_1, \dots, v_n\})$
     $=$ (Lemma A.2.2) $\mathsf{fn}(Q) \setminus \{b_1, \dots, b_n\} \cup \mathsf{fn}(\{v_1, \dots, v_n\})$

   - If $\{\tilde{v}\} \cap \{\tilde{b}\} \neq \emptyset$

     $\mathsf{fn}(P) \overset{\text{def}}{=} \{a\} \cup \mathsf{fn}(Q) \setminus \{b_1 \dots b_n\} \cup \mathsf{fn}(\{v_1 \dots v_n\})$

     $\mathsf{fn}(Q\{v_1/b_1 \dots v_n/b_n\})$
     $\overset{\text{def}}{=} \mathsf{fn}(P[c1/b_1] \dots [cn/b_n][v_1/c1] \dots [v_n/cn])$ where $c1, \dots, cn$ are fresh

$= \text{(Proposition } A.3.2)\ \mathsf{fn}(Q[c1/b_1]\ldots[cn/b_n])[v_1/c1]\ldots[v_n/cn]$
$= \text{(Proposition } A.3.1)\ \mathsf{fn}(Q)[c1/b_1]\ldots[cn/b_n][v_1/c1]\ldots[v_n/cn]$
$= \text{(Lemma A.2.4) } \{m_1[c1/b_1]\ldots[cn/b_n][v_1/c1]\ldots[v_n/cn] \mid m_1 \in \mathsf{fn}(Q)\}\cup$
$\cup\, \mathsf{sites}(\{m_1[c1/b_1]\ldots[cn/b_n][v_1/c1]\ldots[v_n/cn] \mid m_1 \in \mathsf{fn}(Q)\})$

Since by hypothesis $c1, \ldots, cn$ are fresh, $\{c1, \ldots, cn\} \cap \{v_1, \ldots, v_n\} = \emptyset$, and since $\{c1, \ldots, cn\} \subset \mathcal{C}$, the substitutions in the sequence $[v_1/c1]\ldots[v_n/cn]$ don't interfere with one another. Therefore,

$\subseteq \{m_1[c1/b_1]\ldots[cn/b_n] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{c1, \ldots, cn\} \cup \{v_1, \ldots, v_n\}\cup$
$\cup\, \mathsf{sites}(\{m_1[c1/b_1]\ldots[cn/b_n] \mid m_1 \in \mathsf{fn}(Q)\} \setminus \{c1, \ldots, cn\} \cup \{v_1, \ldots, v_n\})$

Since by hypothesis $c1, \ldots, cn$ are fresh, $\{c1, \ldots, cn\} \cap \{b_1, \ldots, b_n\} = \emptyset$, and because $\{c1, \ldots, cn\} \subset \mathcal{C}$, the substitutions in the sequence $[c1/b_1]\ldots[cn/b_n]$ don't interfere with one another. Therefore,

$\subseteq (\mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \{c1, \ldots, cn\}) \setminus \{c1, \ldots, cn\} \cup \{v_1, \ldots, v_n\}\cup$
$\cup\, \mathsf{sites}((\mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \{c1, \ldots, cn\}) \setminus \{c1, \ldots, cn\} \cup \{v_1, \ldots, v_n\})$

once again, since $\{c1, \ldots, cn\}$ are fresh, $\{c1, \ldots, cn\} \cap (\{b_1, \ldots, b_n\} \cup \mathsf{fn}(Q)) = \emptyset$,

$\subseteq \mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \{v_1, \ldots, v_n\} \cup \mathsf{sites}(\mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \{v_1, \ldots, v_n\})$

Since $\forall v : v \in \mathsf{fn}(v)$

$\subseteq \mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \mathsf{fn}(\{v_1, \ldots, v_n\}) \cup \mathsf{sites}(\mathsf{fn}(Q)) \setminus \{b_1, \ldots, b_n\} \cup \{v_1, \ldots, v_n\})$
$= \text{(Lemma A.2.2)}\ \mathsf{fn}(Q) \setminus \{b_1, \ldots, b_n\} \cup \mathsf{fn}(\{v_1, \ldots, v_n\})$

This result is also true for the process derivation rules [the induction hypothesis (i.h.) is $\mathsf{fn}(P) \supseteq \mathsf{fn}(Q)$].

[RP-CONT]

$$\frac{P \to Q}{E[P] \to E[Q]}$$

Since P and Q are inserted in the same context $E[]$, and since $E[]$ performs the same changes over the sets $\mathsf{fn}(P)$ and $\mathsf{fn}(Q)$ (i.e., adds or subtracts the same sets of names), the $\supseteq$ relation is preserved.

[RP-STR]

$$\frac{P \equiv P' \quad P \to Q \quad Q' \equiv Q}{P' \to Q'}$$

By Proposition A.6.1, $\mathsf{fn}(P') = \mathsf{fn}(P) \supseteq$ (i.h.) $\mathsf{fn}(Q) = \mathsf{fn}(Q')$.

2. Networks. By induction on the structure of the derivation of $N \rightarrow M$. This result is true for the network axioms:

   [RN-MIGO]

   $$s[a@r!\langle\widetilde{w}\rangle] \rightarrow r[\sigma(a@r!\langle\widetilde{w}\rangle, s, r)] \overset{\text{def}}{=} r[a!\langle\sigma(\widetilde{w}, s, r)\rangle]$$

   $\mathsf{fn}(s[a@r!\langle\widetilde{w}\rangle])$
   $\overset{\text{def}}{=}\mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@s \cup \{s\}$
   $\overset{\text{def}}{=}(\mathsf{fn}(a@r) \cup \mathsf{fn}(\widetilde{w}))@s \cup \{s\}$
   $\overset{\text{def}}{=}\{a@r, r\} \cup \mathsf{fn}(\widetilde{w})@s \cup \{s\}$

   $\mathsf{fn}(r[\sigma(a@r!\langle\widetilde{w}\rangle, s, r)])$
   $\overset{\text{def}}{=}\sigma(\mathsf{fn}((a@r!\langle\widetilde{w}\rangle), s, r)@r \cup \{r\}$
   $\overset{\text{def}}{=}\mathsf{fn}((a@r!\langle\widetilde{w}\rangle)[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r])@r \cup \{r\}$, where (*)
   $\subseteq$ (Proposition $A.3.3$) $(\mathsf{fn}(a@r!\langle\widetilde{w}\rangle)[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r])@r \cup \{r\}$, where (*)

   Inside the two sequences of substitutions $[\tilde{b}@s/\tilde{b}]$ and $[\tilde{a}/\tilde{a}@r]$ there are no interferences, because in each pair of names which constitutes a substitution, an element of $\mathcal{C}$ is being substituted by one from $\mathcal{C}@\mathcal{S}$. Besides this, the the substitutions from the sequence $[\tilde{a}/\tilde{a}@r]$ do not interfere with those in $[\tilde{b}@s/\tilde{b}]$, because they only replace variables belonging to $\mathcal{C}$, while the result from the second belong to $\mathcal{C}@\mathcal{S}$.
   $\subseteq (\mathsf{fn}(a@r!\langle\widetilde{w}\rangle) \setminus \{\tilde{b}, \tilde{a}@r\} \cup \{\tilde{b}@s, \tilde{a}\})@r \cup \{r\}$
   $= (\mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@r) \setminus \{\tilde{b}@r, \tilde{a}@r\} \cup \{\tilde{b}@s, \tilde{a}@r\} \cup \{r\}$

   Now it suffices to note that:

   - By definition of $\{b_1, \ldots, b_m\}$, if $b \in \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)$ then $b \in \{b_1, \ldots, b_m\}$, therefore
     $(\mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@r) \setminus \{\tilde{b}@r, \tilde{a}@r\}$
     $= \mathsf{fn}(a@r!\langle\widetilde{w}\rangle) \setminus \{\tilde{b}, \tilde{a}@r\}$, and $\mathsf{fn}(a@r!\langle\widetilde{w}\rangle) \setminus \{\tilde{b}, \tilde{a}@r\}$
     $= (\mathsf{fn}(a@r!\langle\widetilde{w}\rangle) \setminus \{\tilde{b}, \tilde{a}@r\})@s$
     $\subseteq \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@s$

   - By definition $\{b_1, \ldots, b_m\} \subseteq \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)$, then $\{b_1, \ldots, b_m\}@s$
     $= \{b_1@s, \ldots, b@sm\}$
     $\subseteq \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@s$.

- By definition $\{a_1@r, \ldots, a_n@r\} \subseteq \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)$, then $\{a_1@r, \ldots, a_n@r\}$ $\subseteq \mathsf{fn}(a@r!\langle\widetilde{w}\rangle)@s$.

- $\{r\}\{a@r, r\}$

(*) $[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r] \overset{\text{abv}}{=} [b_1@s/b_1]\ldots[b_m@s/b_m][a_1/a_1@r]\ldots[a_n/a_n@r]$ and $a_i@r, b_j \in \mathsf{fn}(P)$ where $i = 1, \ldots, nj = 1, \ldots, m$

[RN-MIGI]

$$s[a@r?(\tilde{b})Q] \to r[\sigma(a@r?(\tilde{b})Q, s, r)]$$

$\mathsf{fn}(s[a@r?(\tilde{b})Q])$
$\overset{\text{def}}{=} \mathsf{fn}(a@r?(\tilde{b})Q)@s \cup \{s\}$
$\overset{\text{def}}{=} (\mathsf{fn}(a@r) \cup \mathsf{fn}(Q) \setminus \{\tilde{b}\})@s \cup \{s\}$
$\overset{\text{def}}{=} \{a@r, r\} \cup (\mathsf{fn}(Q) \setminus \{\tilde{b}\})@s \cup \{s\}$

$\mathsf{fn}(r[\sigma(a@r?(\tilde{b})Q, s, r)])$
$\overset{\text{def}}{=} \mathsf{fn}(\sigma(a@r?(\tilde{b})Q, s, r))@r \cup \{r\}$
$\overset{\text{def}}{=} \mathsf{fn}((a@r?(\tilde{b})Q)[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r])@r \cup \{r\}$, with $(*)$
$\subseteq$ (Proposition $A.3.3$) $(\mathsf{fn}(a@r?(\tilde{b})Q)[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r])@r \cup \{r\}$, with $(*)$

Inside the two sequences of substitutions $[\tilde{b}@s/\tilde{b}]$ and $[\tilde{a}/\tilde{a}@r]$ there are no interferences, because in each pair of names which constitutes a substitution, an element of $\mathcal{C}$ is being substituted by one from $\mathcal{C}@\mathcal{S}$. Besides this, the the substitutions from the sequence $[\tilde{a}/\tilde{a}@r]$ do not interfere with those in $[\tilde{b}@s/\tilde{b}]$, because they only replace variables belonging to $\mathcal{C}$, while the result from the second belong to $\mathcal{C}@\mathcal{S}$.
$\subseteq (\mathsf{fn}(a@r?(\tilde{b})Q) \setminus \{\tilde{b}, \tilde{a}@r\} \cup \{\tilde{b}@s, \tilde{a}\})@r \cup \{r\}$
$= (\mathsf{fn}(a@r?(\tilde{b})Q)@r) \setminus \{\tilde{b}@r, \tilde{a}@r\} \cup \{\tilde{b}@s, \tilde{a}@r\} \cup \{r\}$

Now it suffices to note that:

- By definition of $\{b_1, \ldots, b_m\}$, if $b \in \mathsf{fn}(a@r?(\tilde{b})Q)$ then $b \in \{b_1, \ldots, b_m\}$, therefore
  $(\mathsf{fn}(a@r?(\tilde{b})Q)@r) \setminus \{\tilde{b}@r, \tilde{a}@r\}$
  $= \mathsf{fn}(a@r?(\tilde{b})Q) \setminus \{\tilde{b}, \tilde{a}@r\}$, and $\mathsf{fn}(a@r?(\tilde{b})Q) \setminus \{\tilde{b}, \tilde{a}@r\}$
  $= (\mathsf{fn}(a@r?(\tilde{b})Q) \setminus \{\tilde{b}, \tilde{a}@r\})@s$
  $\subseteq \mathsf{fn}(a@r?(\tilde{b})Q)@s$

- By definition $\{b_1, \ldots, b_m\} \subseteq \mathsf{fn}(a@r?(\tilde{b})Q)$, therefore

$$\{b_1, \ldots, b_m\}@s = \{b_1@s, \ldots, b_m@s\} \subseteq \mathsf{fn}(a@r?(\tilde{b})Q)@s.$$

- By definition $\{a_1@r, \ldots, a_n@r\} \subseteq \mathsf{fn}(a@r?(\tilde{b})Q)$, therefore

  $$\{a_1@r, \ldots, a_n@r\} \subseteq \mathsf{fn}(a@r?(\tilde{b})Q)@s.$$

- By definition $\{r\}\{a@r, r\}$.

(*) $[\tilde{b}@s/\tilde{b}][\tilde{a}/\tilde{a}@r] \overset{\text{abv}}{=} [b_1@s/b_1] \ldots [b_m@s/b_m][a_1/a_1@r] \ldots [a_n/a_n@r]$ and $a_i@r, b_j \in \mathsf{fn}(P)$ where $i = 1, \ldots, n$ and $j = 1, \ldots, m$

[RN-CONT]

$$\frac{N \to M}{F[N] \to F[M]}$$

Since N and M are inserted in the same context F, and since F this context defines the same changes over the sets $\mathsf{fn}(N)$ and $\mathsf{fn}(M)$ (i.e., it adds or subtracts the same sets of names), the $\supseteq$ relation is preserved.

[RN-STR]

$$\frac{N \equiv N' \quad N \to M \quad M' \equiv M}{N' \to M'}$$

By Proposition A.6.1, $\mathsf{fn}(N') = \mathsf{fn}(N) \supseteq$ (i.h.) $\mathsf{fn}(M) = \mathsf{fn}(M')$.

[RN-SITE]

$$\frac{P \to Q}{s[P] \to s[Q]}$$

$\mathsf{fn}(s[P]) = \mathsf{fn}(P)@s \supseteq$ (i.h.)$\mathsf{fn}(Q)@s = \mathsf{fn}(s[Q])$.

□

Proof of Proposition A.7.2. Analogously to the previous proof, this proof consists in an induction on the derivation. Proposition A.4.2 is used. Proposition A.7.1 is useful for proving the case of CONT.

1. Processes. By induction over the structure of the derivation of $P \to Q$. This result is true for the process axioms:

   [RP-COMM]

   $$P = (a?(\tilde{b})Q \mid a!\langle\tilde{v}\rangle); \quad P \to Q\{\tilde{v}/\tilde{b}\}$$

By hypothesis P ok, thus, Q ok. If $\{\tilde{v}\} \cap \{\tilde{b}\} = \emptyset$, then $Q\{\tilde{v}/\tilde{b}\} \overset{\text{def}}{=} Q[v_1/b_1]\ldots$ $[v_n/b_n]$. $[v_1/b_1]\ldots[v_n/b_n]$ is a finite sequence of name instantiations , therefore by Proposition A.4.2 $Q\tilde{v}/\tilde{b}$ ok. If, on the contrary, $\{\tilde{v}\} \cap \{\tilde{b}\} \neq \emptyset$, we have that $Q\{\tilde{v}/\tilde{b}\} \overset{\text{def}}{=} Q[w_1/b_1]\ldots[w_n/b_n][v_1/w_1]\ldots[v_n/w_n]$ where w_i are fresh. Then, since $[w_1/b_1]\ldots$
$[w_n/b_n]$ is a finite sequence of name replacements, by Proposition A.4.1 $Q[w_1/b_1]\ldots$ $[w_n/b_n]$ ok. In turn, since $[v_1/w_1]\ldots[v_n/w_n]$ is a finite sequence of name instantiations, by Proposition A.4.2
$Q[w_1/b_1]\ldots[w_n/b_n][v_1/w_1]\ldots[v_n/w_n]$ ok.

This is also true for the process derivation rules:

[RP-CONT]

$$\frac{P \to Q}{E[P] \to E[Q]}$$

In order to simplify the proof, without loss of the strength of the proof, we only analyze the contexts E where $E1 = []$, for it is possible to derive the same reductions using the [RP-CONT] rule repeatedly.

- $E = []$

  By the induction hypothesis, $E[P] = P$ verifies the syntactic restrictions, therefore the same happens with $E[Q]$.

- $E = ([] \mid P_1)$

  $E[P]$ ok by hypothesis, and by definition we conclude that P ok and P_1 ok. Therefore, $P \mid P_1$ ok.

- $E = ((\nu\,)\,[])$

  $E[P]$ ok by hypothesis, therefore P ok, and by definition if $E[P]$ ok, we have two cases:

  – If $n = a$, $\forall t : a@t \notin \mathsf{fn}(P)$. By Proposition A.7.1 $\mathsf{fn}(P) \supseteq \mathsf{fn}(Q)$, therefore $\forall t : a@t \notin \mathsf{fn}(Q)$. It follows that $E[P]$ ok.

  – If $n = a@s$, $a \notin \mathsf{fn}(P)$. By Proposition A.7.1 $\mathsf{fn}(P) \supseteq \mathsf{fn}(Q)$, therefore $a \notin \mathsf{fn}(Q)$. It follows that $E[P]$ ok.

[RP-STR]

$$\frac{P \equiv P' \quad P \to Q \quad Q' \equiv Q}{P' \to Q'}$$

Suppose that P' ok; then, by Proposition A.6.2 we also have that P ok, and by i.h. Q ok, and finally by Proposition A.6.2 we have that Q' ok.

2. Networks. This result is true for the network axioms:

   [RN-MIGO]

   $$N = s[a@r!\langle\widetilde{n}\rangle]; \quad N \to r[\sigma(a@r!\langle\widetilde{n}\rangle, s, r)]$$

   Suppose that N ok, then by definition $a@r!\langle\widetilde{n}\rangle$ ok. By definition, $\sigma(a@r!\langle\widetilde{n}\rangle, s, r) \overset{\text{def}}{=} (a@r!\langle\widetilde{n}\rangle SEQ$, where SEQ is a finite sequence of name translations. Therefore, by Proposition A.4.3 , $(a@r! < nn >)$SEQ ok, thus N ok.

   [RN-MIGI]

   $$N = s[a@r?(\tilde{b})Q]; \quad N \to r[\sigma(a@r?(\tilde{b})Q, s, r)]$$

   Suppose that N ok, then by definition $a@r?(\tilde{b})Q$ ok. By definition, $\sigma(a@r?(\tilde{b})Q, s, r) \overset{\text{def}}{=} (a@r?(\tilde{b})Q)$SEQ, where SEQ is a finite sequence of name translations. Therefore, by Proposition A.4.3 , $(a@r?(bb).Q)$SEQ ok, thus N ok.

   This result is also true for the network derivation rules:

   [RN-CONT]

   $$\frac{N \to M}{F[N] \to F[M]}$$

   [The induction hypothesis 1 (i.h.1.) is $N \to M$ and N ok $\Rightarrow M$ ok]
   In order to simplify the proof, without loss of the strength of the proof, we only analyze the contexts $F[]$ where $F1[] = []$, for it is possible to derive the same reductions using the [RN-CONT] rule repeatedly.

   - $F = []$

     By the induction hypothesis, $F[P] = P$ verifies the syntactic restrictions, therefore the same happens with $F[Q]$.

   - $F = ([] \mid P_1)$

$F[P]$ ok by hypothesis, and by definition we have that P ok and P_1 ok. Therefore, $P \mid P_1$ ok.

- $F = ((\nu\, g)\, [])$

  $F[P]$ ok by hypothesis, therefore P ok, and by definition if $F[P]$ ok, then $F[P]$ ok.

[RN-STR]

$$\frac{N \equiv N' \quad N \to M \quad M' \equiv M}{N' \to M'}$$

Suppose that N' ok; then, by Proposition A.6.2 we also have that N ok, and by i.h. M ok. Finally, by Proposition A.6.2 we have that M' ok.

[RN-SITE]

$$\frac{P \to Q}{s[P] \to s[Q]}$$

Suppose that $s[P]$ ok. Then P ok, by i.h. Q ok, and by definition $s[Q]$ ok.

□

A.2 Results on the type system

Lemma A.8 (Free and bound names). If $\Gamma \vdash_s P$ then $(\mathsf{fn}(P) \cap \mathcal{C}) \subseteq \mathrm{dom}(\Gamma(s))$ and $(\mathsf{bn}(P) \cap \mathcal{C}) \cap \mathrm{dom}(\Gamma(s)) = \emptyset$.

Proof. By induction on the derivation of the judgment. □

Lemma A.9 (Substitution). Let $\Gamma \vdash_s P$. Then:

1. $\Gamma[t/r] \vdash_s P[t/r]$; and
2. if $x \in \mathsf{fn}(P)$ but $x@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$, then $\Gamma[a@r/x@s] \vdash_s P[a@r/x]$ for all r, s and moreover $\Gamma[a@s/x@s] \vdash_s P[a/x]$ when $r = s$.

Proof. The first clause is proved by a simple induction on the derivation of the judgment. The second is proved by induction on the derivation of the judgment $\Gamma[a@r/x@s] \vdash_s P[a@r/x]$.

Base case: $P = u!\langle\tilde{v}\rangle$.

Notice that x cannot occur both in u and in $\tilde{v}$, as by hypothesis P is typable (and clearly, without recursive types we are not able to type processes like $x!\langle x\rangle$). Two cases should be considered:

1. Case $u = x$.
   By hypothesis $\Gamma \overset{\text{def}}{=} \Gamma' \uplus \{s{:}\{x{:}\gamma\}\} \vdash_s x!\langle\rangle = P$, and since $\Gamma[a@r/x@s]$ is defined, $\Gamma \vdash_s a@r{:}\gamma$; it follows that:
   (a) by applying Definitions 3.2 and 4.7, and rule TP-OUTL, one gets
   $$\Gamma[a@r/x@s] \overset{\text{def}}{=} \Gamma'' \uplus \{r{:}\{a{:}\gamma\}\} \vdash_s a@r!\langle\rangle = P[a@r/x]\,;$$
   (b) moreover, if $r = s$, by applying Definitions 3.2 and 4.7, and the rules TP-OUTL and TP-OUTS, one gets
   $$\Gamma[a@s/x@s] \overset{\text{def}}{=} \Gamma'' \uplus \{s{:}\{a{:}\gamma\}\} \vdash_s a!\langle\rangle = P[a/x]\,.$$
   We attained the envisaged result in both cases.
2. Consider now the case $x \in \{\tilde{v}\}$:
   By axioms TS-LCH and TS-SCH, making $\Delta \overset{\text{def}}{=} \{s{:}\{x{:}\gamma\}\}$, then $\Delta \vdash_s x{:}\gamma$ and $\Delta[a@r/x@s] \vdash_s a@r{:}\gamma$.
   (a) If $v = x$ then using the fact just referred, proceed like above;
   (b) If x occurs in $\tilde{v}$ only once, then use the fact referred above, rule TS-UNI, and proceed like above;.
   (c) If x occurs in $\tilde{v}$ more than once, remember that the substitution of names in a process is simultaneous; thus, the result follows as described before.

Induction step: The interesting cases are those involving binders.

1. Let $P = c@t?(\tilde{x})Q$.

   Making $\Gamma' \overset{\text{def}}{=} \Gamma'' + \{t{:}\{c{:}\mathit{Ch}(\widetilde{\gamma})\}\}$, the hypothesis of rule TP-Inpl ensures that

   $$\Gamma'' \uplus \{s{:}\{x{:}\gamma\}\} \uplus \{s{:}\{\widetilde{x{:}\gamma}\}\} \vdash_s Q\,.$$

   Furthermore, since by hypothesis $x@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$,

   $$(c@t)[a@r/x] = c@t = (c@t)[a/x]\,.$$

   (a) We examine first the result of $P[a@r/x]$, according to Definition 3.2: since $x \in \mathsf{fn}(P)$, then $x \notin \{\tilde{x}\}$; as clearly $x@s \notin \{\tilde{x}\}$ and $a@r \notin \{\tilde{x}\}$,

   $$(c@t?(\tilde{x})Q)[a@r/x] = c@t?(\tilde{x})Q[a@r/x]\,,$$

   and the result follows using the induction hypothesis and the referred rule.

   (b) In the case of $P[a/x]$, there are two alternatives:

      i. If $a \notin \{\tilde{x}\}$, proceed as above.

      ii. Otherwise, for some fresh y,

      $$(c@t?(x_1 \cdots a \cdots x_n)Q)[a/x] = c@t?(x_1 \cdots y \cdots x_n)Q[y/a][a/x]\,.$$

      Notice that the Lemma A.8 ensures that $\{\tilde{x}\} \cap \mathrm{dom}(\Gamma(s)) = \emptyset$; it is possible to choose y such that $\Gamma \vdash_s c@t?(x_1 \cdots y \cdots x_n)Q[y/a]$. Thus, the result follows using the induction hypothesis and rule TP-Inpl.

2. Let $P = c?(\tilde{x})Q$.

   The last rule one applies to derive the judgment is TP-Inps. In this case one cannot use the induction hypothesis directly with the premise of this rule, but should use it with the premise of TP-Inpl (the premise of the premise). Then, the result follows easily.

3. Let $P = (\nu\, t)\, Q$.

   Making $\Gamma' = \Gamma'' \uplus \{t{:}\varphi\}$, the hypothesis of rule TP-Resn ensures that

   $$\Gamma = \Gamma'' \uplus \{t{:}\varphi\} \uplus \{s{:}\{x{:}\gamma\}\} \vdash_s Q\,.$$

   We examine $((\nu\, t)\, Q)[a@r/x]$ according to Definition 3.2:

   (a) If $t \neq r$ then $((\nu\, t)\, Q)[a@r/x] = (\nu\, t)\, Q[a@r/x]$ and the result follows using the induction hypothesis.

   (b) Otherwise, for some fresh t' we have

   $$((\nu\, t)\, Q)[a@r/x] = (\nu\, t')\, Q[t'/t][a@r/x],$$

   and the result follows using the the first clause of this lemma and the induction hypothesis.

   For $((\nu\, t)\, Q)[a/x]$ the reasoning is similar

4. Let $P = (\nu\, c@t)\, Q$.

   As before, we examine $((\nu\, c@t)\, Q)[a@r/x]$ and $((\nu\, c@t)\, Q)[a/x]$ according to Definition 3.2. Notice first that if $c = x$ then $P = P[a@r/x]$ and by the Lemma A.8 also $\Gamma[a@r/x@s] = \Gamma[a@r/x@s]$; thus the result is, in this case, trivial. For the remaining cases of the substitution the reasoning is similar to that done in the case above.

5. Let $P = (\nu\, c)\, Q$. The reasoning is similar to that done in the previous case.

The remaining cases are straightforward.

$\square$

Corollary A.10 (Simultaneous substitution). Let $\Gamma \vdash_s P$, and consider that $x \in \mathsf{fn}(P)$ but $x@\mathcal{S} \cap \mathsf{fn}(P) = \emptyset$. Then, $\Gamma\{\widetilde{v@s}/\widetilde{x@s}\} \vdash_s P\{\tilde{v}/\tilde{x}\}$.

Proof. By induction on the length of $\tilde{v}$, using the previous result.

Base case: Let $\Gamma \uplus \{s{:}\{x{:}\gamma\}\} \vdash_s P$ and $\Gamma \vdash_s v{:}\gamma$.

1. Case $v = a@r$.

   Notice that $v@s = v$; the results follows directly from the first case of the second clause of the previous lemma.

2. Case $v = a$.

   The results follows directly from the second case of the second clause of the previous lemma.

Induction step: Let $\Gamma \uplus \{s{:}\{x\tilde{x}{:}\gamma\widetilde{\gamma}\}\} \vdash_s P$ and $\Gamma \vdash_s v\tilde{v}{:}\gamma\widetilde{\gamma}$.

Recall that in $x\tilde{x}$ elements are pairwise disjoint. Thus, the definitions of simultaneous substitutions (Definition 3.12 and its version for typings in page 23) allow the following rearrangements:

$$P\{v\tilde{v}/x\tilde{x}\} = P\{\tilde{v}/\tilde{x}\}[v/x]\,, \text{ and}$$

$$\begin{array}{ll}
(\Gamma \uplus \{s{:}\{x\tilde{x}{:}\gamma\widetilde{\gamma}\}\})\{v@s\widetilde{v@s}/x@s\widetilde{x@s}\} & = \\
((\Gamma \uplus \{s{:}\{x{:}\gamma\}\} \uplus \{s{:}\{\widetilde{x{:}\gamma}\}\})\{\widetilde{v@s}/\widetilde{x@s}\})[v@s/x@s] & = \\
((\Gamma \uplus \{s{:}\{\widetilde{x{:}\gamma}\}\})\{\widetilde{v@s}/\widetilde{x@s}\} \uplus \{s{:}\{x{:}\gamma\}\})[v@s/x@s]\,. &
\end{array}$$

The result follows using the induction hypothesis and doing a case analysis on v, just as for the induction base.

$\square$