MiKO: Mikado Koncurrent Objects*

Francisco Martins† Liliana Salvador‡ Vasco T. Vasconcelos§
Luís Lopes†

May 12, 2005

Abstract

We present MiKO, a distributed process calculus obtained by instantiating Boudol's Generic Membrane Model with the TyCO language. In the membrane model, a network is composed of a collection of domains, each of which comprises an outer part, the *membrane*, and an inner part, the *contents*. The contents of the domain is its computational core. It interacts with the outside (the other domains in the network) using the membrane as intermediary. The membrane implements all protocols required to control the flow of information between the network and the contents of the domain. We provide an operational semantics and a type system for the calculus and prove subject reduction, together with some examples.

1 Introduction

Process calculi provide powerful abstractions and an underlying theory to reason about concurrent, communication based systems. In the last decade there has been an increasing interest in using such calculi to model distributed systems, particularly in the presence of code or computation mobility. Allowing resources to move between the nodes of a distributed system introduces new, non-trivial, problems (e.g., binding policy, marshaling, security) that must be addressed by the underlying process calculus and associated theory.

There have been many proposals of calculi to describe distributed mobile systems. Most of the proposals came from the π-calculus community [19], namely Dπ [15], π_{1l} [1], $lsd\pi$ [21], the Seal calculus [11], Nomadic Pict [30], and the Distributed Join calculus [13], or from the Ambient calculus community [10], notably boxed ambients [6] and safe ambients [16]. Other works comprise, for instance, KLAIM [12], based on the LINDA model.

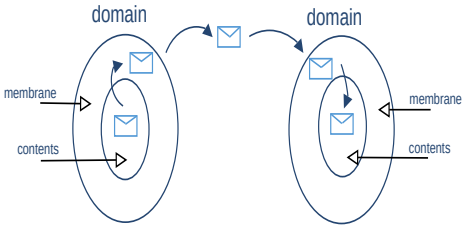
Common to all these works is the concept of computational area, typically a named location where processes run, generically known as a *domain*. Another concept global to these calculi is that of *mobility*, where moving entities range from processes to entire hierarchies of domains. The control of mobility is, in all cases, somehow limited. For instance, in π_{1l} code mobility depends on the state of the domain, and in $lsd\pi$, $D\pi$, and the Ambient calculus mobility might be controlled by means of a type system [8,

*Work funded by EU-FET on Global Computing, project MIKADO IST–2001–32222.
†Universidade dos Açores, Departamento de Matemática
‡Universidade do Porto, Departamento de Ciência de Computadores
§Universidade de Lisboa, Departamento de Informática

9, 17, 18]. Still, it is not easy to program, for instance, the authentication of a client's certificate or the forwarding of incoming messages to a third-party destination.

In the present work we try to overcome this problem by building on the notion of programmable membrane, as introduced by Boudol [3, 4, 5]. Membranes are not explicitly present in any of the works mentioned above and, as far as we are aware, has been explored in the M-calculus [23], in the calculus of Kells [24], in the Brane calculi [7], and in [14].

The key idea behind the membrane model is that each domain is bipartite into a computing body and a membrane. All exchanges between the interior and the exterior of the domain must go through the membrane for validation. The following diagram represents the process of sending a message between two domains, using the membranes of each domain as intermediates.



The model is parametric on the computational structure of the membranes, that is, the calculus that is used to implement the membrane protocols. Thus it is possible to combine the membrane model with your favorite process calculus to obtain a fully functional model for domain-based distributed computing.

In this context we present and instance of the membrane model, using the TyCO-calculus [29] to implement the behavior of both the membranes and the contents of domains. TyCO is a form of the π-calculus that features built-in objects that may respond to one of several method calls, as opposed to simple inputs as in the π-calculus. TyCO's builtin objects are exactly what is needed to provide for the various services provided by the membrane interface.

At the programming level, our goal is to implement membranes as modules for which the only relevant information is the interface they provide. This approach allows to develop *libraries* of membranes with distinct behaviors that may readily be used to implement applications. Thus, ideally, to program a domain one would first choose an appropriate membrane from the library and then implement the application specific contents of the domain.

Outline of the paper. The remainder of this paper is structured as follows: sections 2 to 4 describe the syntax, the semantics, and a type system for MiKO. Section 5 presents examples aiming at providing a better understanding of the capabilities of the language. Section 6 compares work in the area. The last section concludes the paper. Detailed proofs of the results are found in the appendices.

2 Syntax

This section introduces the syntax of MiKO.

We rely on a set of names $\mathcal{N}$, and on a set of labels $\mathcal{L}$, disjoint from $\mathcal{N}$. Names are used to describe *domains* and *channels*; to improve readability we use r, s, t to range

$$
\begin{array}{rcll}
N & ::= & \mathsf{inaction} \mid x\{m\rangle P\}[P] \mid x\,!\,M \mid N \mid N \mid \mathsf{new}\ x\ N & \text{(networks)} \\
P & ::= & \mathsf{inaction} \mid x\,!\,M \mid x?m \mid x?{*}m \mid P \mid P \mid \mathsf{new}\ x\ P \mid & \text{(processes)} \\
 & & A\vec{V} \mid \mathsf{in}[P] \mid \mathsf{out}[x, M] \mid \mathsf{mkdom}[x, m\rangle P, P]\ \mathsf{in}\ P & \\
M & ::= & l[\vec{V}] & \text{(messages)} \\
m & ::= & \{l_i = A_i\}_{i \in I} & \text{(methods)} \\
A & ::= & (\vec{x})P & \text{(abstractions)} \\
V & ::= & x \mid A & \text{(values)}
\end{array}
$$

Figure 1: Syntax of MiKO

over domain names, a, b, c to range over channel names, and x, y, z to range over both domain and channel names. Figure 1 describes the syntax of the calculus, where $\vec{x}$ describes a sequence of names $x_1 \ldots x_n$ ($n > 0$), and similarly for values.

MiKO is an higher-order process calculus, manipulating values that can be either names x (channels or sites) or processes abstracted on a sequence of names $(\vec{x})P$. The calculus is organized in two layers: *networks* and *processes*. A network consists of a collection of *domains* and *network messages* running in parallel. For processes we use TyCO [26, 27, 29] enriched with three new constructs: **in**, **out**, and **mkdom**.

A domain $x\{m\rangle P\}[Q]$ is a location named x, composed of a *membrane* $m\rangle P$ and a *contents* Q. Membranes define the interaction between the domain and the outside world, as well as between the membrane and its contents Q. The method suite m defines the interface of the domain (with the exterior and the interior), and process P is used to keep the state of the membrane. The contents Q of the domain runs the domain's core business. A network message $x\,!\,l[\vec{V}]$ denotes a message, labeled with l and carrying values $\vec{V}$, targeted at domain x. The remaining network constructs are standard: inaction denotes the terminated network, $N \mid L$ describes the parallel composition of two networks N and L; and new $x\ N$ introduces a new name x in network N.

Processes are used in the membrane and in the contents of domains. Its syntax includes objects $x?m$ reading messages from channel x and processing them according to the methods in m, the invocation $x\,!\,l[\vec{V}]$ of a method l with arguments $\vec{V}$ in an object or a domain x, and the application $A\vec{V}$ of an abstraction A to a sequence of arguments $\vec{V}$. An abstraction is a process P parametrized on a sequence of names $\vec{x}$, written $(\vec{x})P$.

Processes also include three special-purpose constructs: in$[P]$ that launches a process P in the contents part of the domain, out$[x, M]$ that sends a message M to domain x, and mkdom$[x, m\rangle P, Q]$ in R that creates a new domain x with membrane $m\rangle P$ and contents Q, visible in process R.

The following example, inspired in [5], illustrates the syntax of the calculus. Consider a domain with a transparent membrane: a membrane that routes every message in and out of the domain. This membrane offers two methods: enter that accepts a message from the network, and exit that routes a message to the network. Two domains, both equipped with transparent membranes, communicate via a very simple protocol: (1) the source domain issues an exit command; (2) the source's exit method selects the enter method in the target domain; (3) the enter method in the target domain launches

$$\mathsf{new}\ x\ N \mid M \equiv \mathsf{new}\ x\ (N \mid M) \qquad \text{if } x \notin \mathrm{fn}(M) \qquad \text{(SN-SCOPE)}$$
$$s\{m\rangle\mathsf{new}\ c\ S\}[P] \equiv \mathsf{new}\ c\ s\{m\rangle S\}[P] \qquad \text{if } c \notin \mathrm{fn}(s, m, P) \qquad \text{(SN-MEMB)}$$
$$s\{m\rangle S\}[\mathsf{new}\ c\ P] \equiv \mathsf{new}\ c\ s\{m\rangle S\}[P] \qquad \text{if } c \notin \mathrm{fn}(s, m, S) \qquad \text{(SN-CONT)}$$
$$\mathsf{inaction} \equiv \mathsf{new}\ s\ s\{m\rangle\mathsf{inaction}\}[\mathsf{inaction}] \qquad \text{(SN-INACT)}$$
$$\mathsf{new}\ x\ \mathsf{new}\ y\ N \equiv \mathsf{new}\ y\ \mathsf{new}\ x\ N \qquad \text{(SN-EXCH)}$$

Figure 2: Structural congruence on networks

the received message in its contents area. A possible implementation of a transparent membrane is as follows.

```
Transparent =
{
  enter (source, x) = in [x[]]
  exit (target, x) = out [target, enter[x]] {– private –}
}
```

Method exit sends an abstraction to the target domain. Method enter applies the received abstraction x to the empty vector [] (thus obtaining a process), and launches it in the domain contents. The membrane needs no state, as reflected by **inaction** below. The sending of a process P from domain r to domain s may be written as follows.

```
r{Transparent⟩inaction}[r!exit[s, ()P]]
```

To keep the calculus simples, we do not distinguish between methods meant to be used by external (network) and by internal (membrane or contents) messages. Instead, in the examples, we mark the latter kind of methods with a *private* comment.

3 Operational Semantics

This section introduces the operational semantics of MiKO. We define a reduction semantics on networks (Figure 4) and on processes (Figure 5) making use of simple structural congruence relations [19] for networks (Figure 2) and for processes (Figure 3).

Bindings, free names, and substitution. The binding occurrences are the name x in $\mathsf{new}\ x\ N$, in $\mathsf{new}\ x\ P$, in $\mathsf{mkdom}[x, m\rangle S, P]$ in Q, as well as names x_i in $(x_1 \ldots x_n)P$. Free and bound identifiers are defined as usual, and we work up to α-equivalence. The set of free names for networks, processes, and methods is denoted by $\mathrm{fn}(\cdot)$. Substitution on processes, notation $P[V/y]$, denotes the standard capture-avoiding substitution of name y for value V, in process P.[1]

Structural Congruence. The structural congruence relation for networks (respectively, for processes), $\equiv$, is the least congruence relation closed under the rules in

[1] Substitution is not total. For example, $(c!M)[A/c]$ confounds instantiation of abstractions with that of names. Sangiorgi shows that substitution is defined for a certain class of processes, that of well-sorted processes [22]. It should be easy to show that the same property holds for well-typed processes, as defined in Section 4.

$$\mathsf{new}\ c\ P \mid Q \equiv \mathsf{new}\ c\ (P \mid Q) \quad \text{if } c \notin \mathrm{fn}(Q) \qquad \text{(SP-SCOPE)}$$
$$\mathsf{inaction} \equiv \mathsf{new}\ c\ \mathsf{inaction} \qquad \text{(SP-INACT)}$$
$$\mathsf{new}\ a\ \mathsf{new}\ b\ P \equiv \mathsf{new}\ b\ \mathsf{new}\ a\ P \qquad \text{(SP-EXCH)}$$

Figure 3: Structural congruence on processes

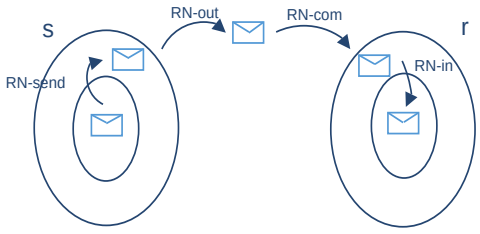
$$s\{m\rangle P\}[s\,!\,M \mid Q] \to s\{m\rangle(m.M \mid P)\}[Q] \qquad \text{(RN-SEND)}$$
$$s\{m\rangle(\mathsf{out}[r, l[\vec{V}]] \mid S)\}[P] \to r\,!\,l[s\vec{V}] \mid s\{m\rangle S\}[P] \qquad \text{(RN-OUT)}$$
$$s\,!\,M \mid s\{m\rangle S\}[P] \to s\{m\rangle(S \mid m.M)\}[P] \qquad \text{(RN-COM)}$$
$$s\{m\rangle(\mathsf{in}[P] \mid S)\}[Q] \to s\{m\rangle S\}[Q \mid P] \qquad \text{(RN-IN)}$$
$$s\{m\rangle(\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T)\}[Q] \to \mathsf{new}\ r\ (r\{m'\rangle S\}[P] \mid s\{m\rangle(R \mid T)\}[Q]) \quad \text{if } r \notin \mathrm{fn}(s, m, T, Q) \qquad \text{(RN-MKD)}$$
$$\frac{S \to T}{s\{m\rangle S\}[P] \to s\{m\rangle T\}[P]} \qquad \frac{P \to Q}{s\{m\rangle S\}[P] \to s\{m\rangle S\}[Q]} \qquad \text{(RN-MEMB,RN-CONT)}$$
$$\frac{N \to N'}{N \mid L \to N' \mid L} \qquad \frac{N \to N'}{\mathsf{new}\ x\ N \to \mathsf{new}\ x\ N'} \qquad \frac{N' \equiv N \quad N \to L \quad L \equiv L'}{N' \to L'} \qquad \text{(RN-PAR,RN-RES,RN-CONG)}$$

Figure 4: Reduction rules on networks

Figure 2 (respectively, in Figure 3) alpha-congruence, and the commutative monoidal rules.

For networks, SN-SCOPE is the usual scope extrusion rule, and SN-MEMB and SN-CONT describe scope extrusion applied to domains. For processes, the rules are standard.

Reduction. The reduction rules for networks and processes are given in Figures 4 and 5, where method invocation $\{l_i = A_i\}_{i \in I}.(l_j[\vec{V}])$ is defined as the process $A_j\vec{V}$, when $j \in I$. The below diagram, depicting migration between domains, explains the first four axioms in Figure 4.



Rule RN-SEND delivers a message $l[\vec{V}]$ from the contents part of the domain to its membrane, calling method l in the domain's interface. The method may then create a process $\mathsf{out}[r, l'[\vec{U}]]$. Rules RN-OUT adds the source domain to the message (thus

$$c?m \mid c!M \to m.M \qquad \text{(RP-COM)}$$

$$c?*m \mid c!M \to c?*m \mid m.M \qquad \text{(RP-REP)}$$

$$((\vec{x})P)\vec{V} \to P[\vec{V}/\vec{x}] \qquad \text{(RP-APP)}$$

$$\frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \to Q}{\mathsf{new}\ c\ P \to \mathsf{new}\ c\ Q} \qquad \frac{P' \equiv P \quad P \to Q \quad Q \equiv Q'}{P' \to Q'}$$

(RP-PAR, RP-RES, RP-CONG)

Figure 5: Reduction rules on processes

$$\begin{array}{rcl} \alpha & ::= & \mathsf{ch}(\rho) \mid \mathsf{dom}(\rho) \mid \mathsf{abs}(\vec{\alpha}) \mid t \mid \mu t.\alpha \\ \rho & ::= & \{l_i : \vec{\alpha_i}\}_{i \in I} \end{array}$$

Figure 6: Syntax of types

obtaining $l'[s\vec{U}]$), and places the message in the network. Rule RN-COM accepts the message from the network, selecting method l' in the interface of the target membrane. The method may then create a process $\mathsf{in}[P]$. Then, rule RN-IN drives process P from the membrane to contents of the domain.

Domains can only be created from the membrane of an existing domain, via rule RN-MKD. To create a domain, we provide a domain name r, a membrane $m'\{\!|S$, the contents P of the domain, and the scope R of the existing domain where the new domain r is visible. Rules RN-MEMB, RN-CONT, RN-PAR, and RN-RES allow reduction inside the membrane, the domain contents, the parallel composition, and name restriction, respectively. Finally, RN-CONG brings structural congruence into reduction.

The rules for processes (Figure 5) are standard [29]

As an example of a reduction in MiKO, consider the domain at the end of Section 2, in parallel with a domain with a transparent membrane and empty contents, where we abbreviate **inaction** to 0.

```
r{Transparent|0}[r!exit[s, ()P]]  |  s{Transparent|0}[0]
```

and using rules RN-SEND, RN-OUT, RN-COM, RN-IN, as described in the diagram above, we successively have:

```
r{Transparent|0}[r!exit[s, ()P]]  |  s{Transparent|0}[0]  →
r{Transparent|out[s,enter[()P]]}[0]  |  s{Transparent|0}[0]  →
r{Transparent|0]}[0]  |  s!enter[r,()P]  |  s{Transparent|0}[0]  →
r{Transparent|0]}[0]  |  s{Transparent|in[P]}[0]  →
r{Transparent|0]}[0]  |  s{Transparent|0}[P]
```

4 The Type System

This section introduces a type system for our language and presents the main results: subject-reduction and type safety.

$$\frac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_s P \mid Q \qquad \Gamma \vdash_\_ s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle P\}[Q]} \quad \text{(TN-DOM)}$$

$$\frac{\Gamma \vdash_s \vec{V}\colon \rho.l \qquad \Gamma \vdash_\_ s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\,!\,l[\vec{V}]} \quad \text{(TN-MSG)}$$

$$\frac{\Gamma \vdash N \qquad \Gamma \vdash L}{\Gamma \vdash N \mid L} \qquad \frac{\Gamma \vdash N \qquad \Gamma \vdash_\_ x \in \{\mathsf{ch}(\_), \mathsf{dom}(\_)\}}{\Gamma \setminus x \vdash \mathsf{new}\ x\ N} \qquad \Gamma \vdash \mathsf{inaction}$$

(TN-PAR,TN-RES,TN-INACT)

Figure 7: Typing networks

Types. We fix a countable set of type variables ranged over by t. The syntax of types is given in Figure 6, where the sequence of types $\alpha_1 \ \ldots \ \alpha_n$ ($n > 0$) is abbreviated to $\vec{\alpha}$.

Types of the form $\mathsf{ch}(\rho)$, $\mathsf{dom}(\rho)$, and $\mathsf{abs}(\vec{\alpha})$ denote types for channels, for domains, and for abstractions, respectively. Records ρ describe a collection of n ($n > 0$) methods labeled l_i accepting arguments of types $\vec{\alpha}_i$. Types are interpreted as rational (regular infinite) trees. A type of the form $\mu t.\alpha$ (with t guarded in α) denotes the rational tree, solution of the equation $t = \alpha$. An interpretation of recursive types as infinite trees naturally induces an equivalence relation $\approx$ on types, by putting $\alpha \approx \beta$ if the underlying trees for α and for β are equal [20].

Typing rules. Typings, denoted by Γ are partial functions of finite domains from names to types. Notation $\Gamma \setminus x$ denotes the typing obtained from Γ by removing x from its domain.

The type system is described in Figures 7–9, where $\{l_i : \vec{\alpha}_i\}_{i \in I}.l_j = \vec{\alpha_j}$ when $j \in I$. It includes four kinds of judgments: (a) judgment $\Gamma \vdash N$ asserts that network N is well typed under typing assumptions Γ; (b) judgment $\Gamma \vdash_s P$ means that process P is well typed under typing assumptions Γ, when running at side s; (c) judgment $\Gamma \vdash_s m\colon \rho$ asserts that the method suite m has type ρ, when located at site s; and (d) judgment $\Gamma \vdash_s V : \alpha$ assigns type α to value V, at site s.

To type a domain $s\{m\rangle S\}[P]$ using rule TN-DOM, one has to type the membrane $m\rangle S$ and the contents P, both in s. Domain s must possess the type of its interface m. To type a message using rule TN-MSG, one must type the contents $\vec{V}$ of the message, and make sure that the type for s is a domain-type whose l-component is the type of $\vec{V}$. To type a name restriction $\mathsf{new}\ x\ N$ using rule TN-RES, we type network N, and check whether the type for name x is a that of a channel or of a site.

When typing processes, we record (under the turnstile) the domain that hosts the process. This piece of information is relevant when typing the **out** operation, since we stamp messages with the name of the sending domain. Rule TP-OBJ expresses the fact that c must be a channel that has a type compatible with method suite m. Rule TP-MSG is similar to TN-MSG, but here x can be a channel or a domain, since we use the same constructor to send messages to membranes or interact with objects. To type the **in** constructor in domain s, we type P in s using rule TP-IN. Rule TP-OUT types the sequence of values $s\vec{V}$ in the target domain r, since abstractions run in the destination domain. Notice that one appends the origin domain s to the list of parameters, as prescribed by the operational semantics (rule RN-OUT in Figure 4). To type domains

$$\frac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\rho)}{\Gamma \vdash_s c?m} \qquad \frac{\Gamma \vdash_s c?m}{\Gamma \vdash_s c?*m} \qquad \text{(TP-OBJ,TP-REP)}$$

$$\frac{\Gamma \vdash_s \vec{V}\colon \rho.l \qquad \Gamma \vdash_{\_} x \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_s x\,!\,l[\vec{V}]} \qquad \text{(TP-MSG)}$$

$$\frac{\Gamma \vdash_s P}{\Gamma \vdash_s \mathsf{in}[P]} \qquad \frac{\Gamma \vdash_r s\vec{V}\colon \rho.l \qquad \Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho)}{\Gamma \vdash_s \mathsf{out}[r, l[\vec{V}]]} \qquad \text{(TP-IN,TP-OUT)}$$

$$\frac{\Gamma \vdash r\{m\rangle S\}[P] \qquad \Gamma \vdash_s R}{\Gamma \setminus r \vdash_s \mathsf{mkdom}[r, m\rangle S, P]\ \mathsf{in}\ R} \qquad \text{(TP-MKD)}$$

$$\frac{\Gamma \vdash_s A\colon \mathsf{abs}(\vec{\alpha}) \qquad \Gamma \vdash_s \vec{V}\colon \vec{\alpha}}{\Gamma \vdash_s A\vec{V}} \qquad \text{(TP-APP)}$$

$$\frac{\Gamma \vdash_s P \quad \Gamma \vdash_s Q}{\Gamma \vdash_s P \mid Q} \qquad \frac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\_)}{\Gamma \setminus c \vdash_s \mathsf{new}\ c\ P} \qquad \Gamma \vdash_s \mathsf{inaction}$$

(TP-PAR,TP-RES,TP-INACT)

Figure 8: Typing processes

$$\frac{\forall i \in I, \quad \Gamma \vdash_s A_i\colon \mathsf{abs}(\vec{\alpha}_i)}{\Gamma \vdash_s \{l_i = A_i\}_{i\in I}\colon \{l_i\colon \vec{\alpha}_i\}_{i\in I}} \qquad \text{(TM-METH)}$$

$$\frac{\Gamma(x) = \alpha \qquad \alpha \approx \beta}{\Gamma \vdash_{\_} x\colon \beta} \qquad \frac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} \vec{x}\colon \vec{\alpha}}{\Gamma \setminus \vec{x} \vdash_s (\vec{x})P\colon \mathsf{abs}(\vec{\alpha})} \qquad \frac{\Gamma \vdash_s V_1\colon \alpha_1 \ \ldots\ \Gamma \vdash_s V_n\colon \alpha_n}{\Gamma \vdash_s V_1 \ldots V_n\colon \vec{\alpha}}$$

(TV-ID,TV-ABS,TV-SEQ)

Figure 9: Typing method suites and values

creation, rule TP-MKD checks that both the created domain $r\{m\rangle S\}[P]$ and its scope R are well-typed. Rule TP-APP checks that the arguments $\vec{V}$ are as expected by the abstraction A. Finally, rule TP-RES is similar to TN-RES, except that only channel restriction is possible in processes.

Rule TV-ID incorporates type equivalence in the type system. The remaining rules should be easy to understand.

Properties. *Subject-Reduction* asserts the invariance of typings under reduction.

Theorem 4.1 (Subject-Reduction) *If* $\Gamma \vdash N$ *and* $N \to L$, *then* $\Gamma \vdash L$.

Type safety asserts that typable programs do not encounter immediate errors. Together with Subject-Reduction we obtain Milner's slogan "Well-typed programs do not go wrong".

Two things can go wrong: a) the invocation of a method non-existent in the target object or domain interface; b) the application of a wrong number of arguments to an abstraction. In the former case $\{l_i = A_i\}_{i\in I}.(l_j[\vec{V}])$ is not defined (that is, $j \notin I$); in the latter $P[V_1 \ldots V_k/x_1 \ldots x_n]$ is not defined (that is, $n \neq k$ or $P[V_i/x_i]$ is not defined for some i). We say that a network is *faulty* if

1. $N \equiv \mathsf{new}\ \vec{x}\ (s\,!\,M \mid s\{m\rangle S\}[P] \mid L)$, and $m.M$ is not defined; or
2. $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle P\}[s\,!\,M \mid Q] \mid L)$, and $m.M$ is not defined; or
3. $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle(c?m \mid c\,!\,M \mid P)\}[Q] \mid L)$, and $m.M$ is not defined, and similarly for $c?*m$; or
4. $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle P\}[c?m \mid c\,!\,M \mid Q] \mid L)$, and $m.M$ is not defined, and similarly for $c?*m$; or
5. $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle(((\vec{y})P)\vec{V} \mid R)\}[Q] \mid L)$, and $P[\vec{V}/\vec{y}]$ is not defined; or
6. $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle P\}[((\vec{y})P)\vec{V} \mid Q] \mid L)$, and $P[\vec{V}/\vec{y}]$ is not defined.

Theorem 4.2 (Type-Safety) *If $\Gamma \vdash N$, then N is not faulty.*

5 Examples

In this section we present several examples to illustrate the use of membranes, as well as the programming style we propose.

5.1 Polarised membrane

Our first example illustrates a membrane that filters messages depending on its internal state. A polarised membrane [1, 5] controls the passage of incoming and outgoing messages using a private channel named membraneStatus. If it is active (active = **true**), then messages flow freely from the network into the contents and vice-versa, as if the membrane were transparent (see section 2). Otherwise (active = **false**) all messages are discarded by the membrane.

Assuming the presence of boolean values and conditional processes, an implementation of the polarised membrane might be:

```
{
  enter(origin, process) =
    membraneStatus?{
      value(active) =
        membraneStatus!value[active] |
        if active then in[process]
    }
  exit(targetDomain, process) =
    membraneStatus?{
      value(active) =
        membraneStatus!value[active] |
        if active then out[targetDomain, enter[process]]
    }
  ping(origin, replyTo) =
    membraneStatus?{
      value(active) =
        membraneStatus!value[active] |
        out[origin, enter[replyTo![active]]]
    }
  deactivate() =                              {- private -}
    membraneStatus?{
```

```
      value(active) =
        membraneStatus!value[false]
    }
  activate() =                                  {– private –}
    membraneStatus?{
      value(active) =
        membraneStatus!value[true]
    }
  askStatus(targetDomain, replyTo) =            {– private –}
    out[targetDomain, ping[replyTo]]
}
⟩ membraneStatus!value[true]
```

If we name the above membrane as polarised, a domain s, initially active, running process P may be written as

```
new membraneStatus s{polarised}[P]
```

The domain state is maintained by a message in transit (initially with value **true**) on channel membraneStatus. Methods *enter* and *exit* check the state of the domain before forwarding the messages to and from the network. Method *deactivate* sets the status of the membrane to **false**.

A domain becomes unavailable when method *deactivate* gets selected. Method ping informs external clients of the current state of the domain, and method askStatus checks whether a target domain that implements the "ping" protocol is active.

5.2 Quality of Service membrane

Our second example illustrates how to program a membrane that guarantees a certain quality of service (QoS) and enforces a protocol between a server and its clients. To assure the required QoS, the server allows only a fixed number of simultaneous requests. Towards this end, it keeps track of the number of concurrent active clients. The state is recorded in a private channel sessionCounter that keeps track of the number of available sessions. The protocol forces a client first to connect to the server, then to repeatedly issue requests (and receive results back), and finally to disconnect from the server.

The server. Channel handler is the link between the membrane and the contents of the server.

The membrane offers four methods: *connect*, *disconnect*, *request*, and *exit*. Method *connect* checks if there are available sessions. If so, it creates a private channel sessionID, sends it to the client (via an **out** operation), interacts with the server's contents (using the **in** operation) to create an object to handle the session, and decrements the session counter. Otherwise, it informs the client that there are too many sessions opened. Method *disconnect* closes the session by selecting the *disconnect* operation on the contents, and incrementing the session counter. Methods *exit* and *request* are as in the transparent membrane (Section 2), only that we have renamed *enter* to *request* for reasons that will become apparent below.

The server's membrane, QoSMembrane, may then be programmed as follows.

```
{
 connect(client, replyTo) =
   sessionCounter?{
```

```
      value(availableSessions) =
        if availableSessions > 0
        then
          new sessionID
          out[client, enter[() replyTo!connected[sessionID]]]|
          in[handler!create[sessionID, client]]
          sessionCounter!value[availableSessions − 1]
        else
          out[client, enter[() replyTo!tooManySessions[]]]|
          sessionCounter!value[availableSessions]
    }
  disconnect(client, sessionID) =
    in[sessionID!disconnect[]] |
    sessionCounter?{
      value(availableSessions) =
        sessionCounter!value[availableSessions + 1]
    }
  request (source, x) = in [x[]]
  exit (target, x) = out [target, enter[x]] {− private −}
}⟫
  sessionCounter![5]
```

On the contents side, the server must provide an implementation for the shared channel handler that handles the various sessions. Each session is associated with a different sessionID. Channel sessionStatus records the availability of the session: a session is available until the client issues a disconnect operation. Results to client requests are first returned to the membrane, which then relays them to the corresponding client domain. The content of a generic server, QoSContents, may be implemented as follows.

```
handler?*{
  create(sessionID, client) =
    new sessionStatus
    sessionStatus!value[true] |
    sessionID?*{
     {− specific server methods −}
     disconnect () =
       sessionStatus?{
         value(alive) =
            sessionStatus!value[false]
       }
    }
}
```

Then, a domain, QoSDomain, running a generic QoS server is the following domain.

```
new sessionCounter
new handler
QoSDomain{QoSMembrane}[QoSContents]
```

The client. The client uses a stateless membrane, and provides a simple method suit, intended to simply follow the protocol with the server. Notice that we can not use the transparent membrane (Section 2) in this case, since the client calls specific methods

on the server (*connect*, *disconnect*, and *request*), rather than the fixed *enter* method provided by the transparent membrane.[2]

The clientMembrane is as follows,

```
{
  enter(server, process) =
    in[process []]
  connect(server, replyTo) =                {– private –}
    out[server, connect[replyTo]]
  disconnect(server, sessionID) =           {– private –}
    out[server, disconnect[sessionID]]
  request(server, computation) =            {– private –}
    out[server, request[computation]]
}
⟩ inaction
```

and, given process clientContents, a client domain becomes:

```
client{clientMembrane}[clientContents]
```

A particular service. As an example of a service the server may implement we use a mathematical server [28] to illustrate the QoS client-server interaction presented above.

For the server side, a math server accepts computation request from clients, compute the request, and reply the results back. We provide the following two methods of the sessionID object, to be added to the part *specific server methods* in the above QoSContents code.

```
add(n, m, replyTo) =
  sessionStatus?{
    value(alive) =
      sessionStatus!value[alive] |
      if alive then
      QoSDomain!exit [client,()replyTo!value[n+m]]
  }
neg (n, replyTo) =
  sessionStatus?{
    value(alive) =
      sessionStatus!value[alive] |
      if alive then
      QoSDomain!exit [client,()replyTo!value[0−n]]
  }
```

Here is the example of a client that (1) establishes a session; (2) asks for the addition two numbers; (3) asks for the symmetric of the result; and finally (4) closes the session.

```
new replyTo
client!connect[QoSDomain, replyTo] |
replyTo?{
  connected(sessionID) =
    new result
    client!request[QoSDomain, ()sessionID!add[3,4,result]]|
    result?{
      value(x) =
```

[2]Since messages are not values, we cannot write a generic method *exit*(server, method) = **out**[server, method].

```
        client!request[QoSDomain, ()sessionID!neg[x,result]]|
        result?{
          value(x) =
            io!printi[x] |
            client!disconnect[QoSDomain, sessionID]
        }
    }
  tooManySessions() =
    io!prints["could not connect to the server"]
}
```

5.3 Members manager

Our last example presents a flavor of programmable membranes that may be reused in several scenarios. The *members manager* provides a controlled access to a domain. Unlike the math server that allows access to any client, we want to give permission to registered users only. So, in order to use the services offered by the domain, the client must first register (thus becoming a *member*) and then authenticate itself with a password. The membrane manages a set of registered users and ensures that only logged users can issue requests to the server. The members membrane may be used in some interesting cases, such as:

- in a *chat server*: only register users can enter the chat services;
- in a *"free" email provider*: users first agree on an email account, setting up a user name and a password, and then access their emails privately;
- in an *e-seller provider*, such as a virtual library or a reserved area in a financial server;
- or even, in a refined *math server*: instead of issuing a connect operation, users first register themselves and then *login* at the server.

Our goal is to implement a *members membrane* in a modular way so that implementing any of the above examples would simply imply writing the service-specific contents, reusing the membrane.

The protocol between a client and the *members membrane* is as follows: the first time the client accesses the server it must register himself. For that purpose, it must provide a tentative *login ID*. If the registration goes well (the login ID is not in use) it gets a password back, otherwise it is notified of the registration failure. Registered users are allowed to engage in a session with the server if the user/password pair they provide matches the information in the list of allowed users. The session proceeds with logged users sending requests to the server and receiving the replies. When leaving the server the client must issue a logout operation to close its session.

The membrane stores the information relative to registered members in a map and the users that are currently logged in a set. In the example below this information is recorded in the membrane state as two private channels named membersMap and loginSet. The membrane offers the following services: *newMember* to register new members; *login* to authenticate users and start a session; *logout* to terminate a session; *request* to accept demands from clients; and *exit* used by the server to deliver answers for client requests.

The outline of the membersMembrane is as follows. We present and comment each method separately, assuming an implementation for maps and for sets.

```
{
  newMember(client, loginName, replyTo) = ...
  login(client, loginName, password, replyTo) = ...
  logout(client, loginName) = ...
  request(client, loginName, job, replyTo) = ...
  exit(client, result) = ...                {– private –}
}
⟩
  memberMap?* { ... } |
  loginSet?* { ... } |
  ...
```

To register a new member we use the following code. [3]

```
newMember(client, loginName, replyTo) =
  if membersMap!containsKey[loginName]
  then
    out[client, enter[replyTo!loginNameExists[]]]
  else
    new password
    membersMap!put[loginName, password] |
    out[client, enter[replyTo!registered[password]]]
```

The implementation is straightforward. We check whether there is a member already registered with the given login name. If not, we generate a password, add the new member to the map, and inform the client of its password. Otherwise, we inform the client of the failed attempt.

Next is a possible implementation of the *login* operation:

```
login(client, loginName, givenPassword, replyTo) =
  if membersMap!containsKey[loginName] and
     membersMap!get[loginName] = givenPassword
  then
    out[client, enter[replyTo!connected[]] |
    loginSet!add[loginName]
  else
    out[client, enter[replyTo!invalidUserPassword[]]]
```

We check whether the client is registered and the password matches. If is everything goes well, we notify the client that the login operation was successful, otherwise we report an error.

The *logout* operation simply updates the login set. The implementation is as follows:

```
logout(client, loginName) =
  loginSet!remove[loginName]
```

There are two generic methods for accepting and replying to requests: the *request* and the *exit* methods. The underlying communication protocol between the client and the server is established via these two methods. The interaction with the contents part of the server is performed via shared channel handler. For instance, a chat server may implement the *ICQ* protocol or a email server the *IMAP* protocol, encapsulating the

[3] TyCO's idiom

if membersMap!*containsKey*[loginName] **then** ...

is short for [25]

new r membersMap!*containsKey*[loginName, r] | r ? {value (x) = **if** x **then** ...

messages using these two methods and providing an implementation for the handler channel. Notice that the membrane checks that the client sending the request is logged in. All the remaining actions are assigned to the contents of the domain. The implementation is simple:

```
request(client, loginName, job, replyTo) =
  if loginSet!contains[loginName]
  then
    in[handler!request[client, loginName, job, replyTo]]
  else
    out[client, enter[replyTo!notLoggedIn[loginName]]]

exit(client, result) =
   out[client, enter[result]]
```

A domain named oscar running an ICQ process (not shown) that implements the *AIM/ICQ* protocol may be set up as:

```
new memberMap
new loginSet
new handler
oscar[membersMembrane]{ICQ}
```

6 Related Work

The introduction mentions a few process-calculi based proposals to describe mobile systems: Dπ [15], π_{1l} [1], $lsd\pi$ [21], the Seal calculus [11], Nomadic Pict [30], variants of the Ambient calculus [10], notably boxed ambients [6] and safe ambients [16], as well as, KLAIM [12]. None of these works, however, encompasses an explicit notion of computational domain. This section concentrates on models of domains with programmable membranes.

The Mikado domain model, as presented by Boudol [3, 4, 5], constitutes the starting point of this work. In fact, MiKO is an instance of the the Mikado's generic domain model.

The M-calculus [23], like MiKO, bases its notion of domain on locations which are composed of a controller and a content process. Unlike MiKO, locations may have sub-locations and thus the network topology is a tree. MiKO networks are flat and communication is point-to-point. In the M-calculus, interaction consists of either messages directed to resources within a remote domain or migration of processes or locations. MiKO provides a more service-oriented view of interaction since only messages directed to methods in the membranes (albeit carrying code) of remote domains are allowed. Moreover, the interface presented to the network by the membrane of a domain is the only data path to the process in the contents of that domain. In fact, this is exactly the same interface that must be used by the contents itself to interact with the network outside. We feel that this symmetric view of membranes provides a clear and modular programming paradigm. In the M-calculus, whenever a message to a resource within a domain gets cleared by predefined filters for incoming messages, it can move freely between the controller and the contents. This distinction in the treatment of outward and inward interaction is, in our view, unnatural as the symmetry of the membrane or controller should be invariant. Migration the M-calculus is based on a "passivation" construct that freezes a process or a domain and sends it to the destination

domain. This is a very powerful feature of the calculus that allows higher order entities to be sent over the network. In MiKO we do not require such a high-level feature as the calculus has abstractions as first-class values and especially because we do not require to migrate domains (our networks are flat).

The brane calculi [7] are used to describe interaction between cells and molecules. The communication between cells is performed through free-floating molecules that bind to the exterior or the interior of the membranes in order to cross them. If we imagine that cells possess specific receptors that bind molecules, then these receptors are the interface of the cell and therefore, the receptors are akin to the labels of our methods. MiKO and the brane calculi have different aims, MiKO being better suited to describe high-level mobile computing.

Gorla, Hennessy, and Sassone [14] present an experiment with membranes that type check the incoming code to certify that the actions it may perform are granted by the target domain. In order to keep the impact of type checking low, they explore a trusting relation between sites and check only the code that arrives from untrusted domains. A domain in MiKO is not able to check the code it receives; the decision of allowing code to pass or not through the membrane is defined only in terms of the source domain, the selected method, and the state of the domain.

7 Conclusion

In this report we presented the MiKO process calculus and associated programming language. MiKO is based on a model for distributed computing where computation happens in domains that are shielded from the remainder of the network by a membrane. The membrane implements whatever protocols are required to forward, discard or filter messages to and from the domain contents.

The original proposal for the domain model [5] makes no assumption on the computational structure (membrane and contents) of domains. MiKO uses the TyCO process calculus [29] to implement both the membrane and the contents of domains. We defined an operational semantics and a type system for the calculus that ensures that well-typed MiKO programs do engage in runtime errors.

The programming language derived from the base calculus is straightforward. Programs are written a in modular way by implementing the membrane and contents as separate components. This allows the setup of general purpose membranes that implement protocols used by the service specific contents of a domain. These components can be effectively reused by applications requiring the same kind of membrane.

Currently the implementation of a compiler and run-time system for MiKO is underway, based on Mikado's software framework for rapid prototyping of run-time systems for mobile calculi [2].

References

[1] R. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proceedings of Coordination'96*, volume 1282 of *LNCS*. Springer, 1997.

[2] L. Bettini, R. De Nicola, D. Falassi, M. Lacoste, L. Lopes, L. Oliveira, H. Paulino, and V. Vasconcelos. A software framework for rapid prototyping of run-time systems for mobile calculi. In *Global Computing*, volume 3267 of *LNCS*, pages 179–207. Springer, 2004.

[3] G. Boudol. Core programming model, release 0. Mikado Deliverable D1.2.0, 2002.

[4] G. Boudol. A parametric model of migration and mobility, release 1. Mikado Deliverable D1.2.1, 2003.

[5] G. Boudol. A generic membrane model. In *Global Computing Workshop*, volume 3267 of *LNCS*, pages 208–222. Springer, 2005.

[6] M. Bugliesi, G. Castagna, and S. Crafa. Access control for mobile agents: The calculus of boxed ambients. *ACM Transactions on Programming Languages and Systems*, 26(1):57–124, 2004.

[7] L. Cardelli. Brane calculi: Interactions of biological membranes. In *Proceedings of Computational Methods in Systems Biology'04*, volume 3082 of *LNCS*, pages 257–280. Springer, 2004.

[8] L. Cardelli, G. Ghelli, and A. Gordon. Mobility types for mobile ambients. In *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 230–239. Springer, 1999.

[9] L. Cardelli, G. Ghelli, and A. Gordon. Ambient groups and mobility types. In *Proceedings of TCS'00*, volume 1872 of *LNCS*, pages 333–347. Springer, 2000.

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

[11] G. Castagna and J. Vitek. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, number 1686 in LNCS, pages 47–77. Springer, 1999.

[12] R. De Nicola, G. Ferrari, and R. Pugliese. Klaim: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[13] C. Fournet, G. Gonthier, J. L´evy, L. Maranget, and D. R´emy. A calculus of mobile agents. In *Proceedings of Concur'96*, LNCS, pages 406–421. Springer, 1996.

[14] D. Gorla, M. Hennessy, and V. Sassone. Security policies as membranes in systems for global computing. In *Proceedings of 3rd EATCS Workshop on Foundations of Global Ubiquitous Computing (FGUC'04)*, ENTCS. Elsevier, 2004.

[15] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Journal of Information and Computation*, 173:82–120, 2002.

[16] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proc. 27th POPL*, pages 352–364. ACM Press, 2000.

[17] F. Martins and A. Ravara. Typing migration control in lsdpi. In *Proceedings of FCS'04*, volume 31, pages 1–12. Turku Centre for Computer Science, 2004.

[18] F. Martins and V. Vasconcelos. History-based access control for distributed processes. In *Proceedings of TGC'05*, LNCS. Springer, 2005. To appear.

[19] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I/II. *Jornal of Information and Computation*, 100:1–77, 1992.

[20] B. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[21] A. Ravara, A. Matos, V. Vasconcelos, and L. Lopes. Lexically scoping distribution: what you see is what you get. In *Proceedings of FGC'03*, volume 85(1) of *ENTCS*. Elsevier, 2003.

[22] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, LFCS, University of Edinburgh, 1993. CST-99-93 (also published as ECS-LFCS-93-266).

[23] A. Schmitt and J.-B. Stefani. The m-calculus: a higher-order distributed process calculus. In *POPL*, pages 50–61, 2003.

[24] J.-B. Stefani. A calculus of kells. In *Proceedings of FGC'03*, volume 85(1). Elsevier, 2003.

[25] V. Vasconcelos. Processes, functions, datatypes. *Theory and Practice of Object Systems*, 5(2):97–110, 1999.

[26] V. Vasconcelos. Core-tyco appendix to the language definition, version 0.2. Technical Report 01–5, Faculty of Sciences of the University of Lisbon, 2001.

[27] V. Vasconcelos and R. Bastos. Core-tyco the language definition, version 0.1. Technical Report 98–3, Faculty of Sciences of the University of Lisbon, 1998.

[28] V. Vasconcelos, A. Ravara, and S. Gay. Session types for functional multithreading. In *Proceedings of Concur'04*, volume 3170 of *LNCS*, pages 497–511. Springer, 2004.

[29] V. Vasconcelos and M. Tokoro. A typing system for a calculus of objects. *1st International Symposium on Object Technologies for Advanced Software*, 742:460–474, 1993.

[30] P. Wojciechowski and P. Sewell. Nomadic pict: Language and infrastructure design for mobile agents. *IEEE Concurrency*, 8(2):42–52, 2000.

A Proof of Subject Reduction

This section presents the proof for Theorem 4.1. The proof uses two main auxiliary results, namely, Substitution (Lemma A.1) and Structural Congruence (Lemma A.4).

Lemma A.1 (Substitution)

1. *If* $\Gamma, x\colon \alpha \vdash_r V\colon \beta$ *and* $\Gamma \vdash_r U\colon \alpha$*, then* $\Gamma \vdash_{r[U/x]} V[U/x]\colon \beta$.
2. *Let* $\rho = \{l_1\colon \vec{\beta_1} \ldots l_n\colon \vec{\beta_n}\}$. *If* $\Gamma, x\colon \alpha \vdash_r m\colon \rho$ *and* $\Gamma \vdash_{\_} U\colon \alpha$*, then* $\Gamma \vdash_{r[U/x]} m[U/x]\colon \rho$.
3. *If* $\Gamma, x\colon \alpha \vdash_r P$ *and* $\Gamma \vdash_{\_} U\colon \alpha$*, then* $\Gamma \vdash_{r[U/x]} P[U/x]$.
4. *If* $\Gamma, x\colon \alpha \vdash N$ *and* $\Gamma \vdash_{\_} U\colon \alpha$*, then* $\Gamma \vdash N[U/x]$.

Proof. The proof of the first three results is by mutual induction on the typing derivation.

Substitution on Values. We proceed by case analysis on the structure of V, tracing the last typing rule applied.

- Case V is y
  By hypothesis we derive

  $$\frac{\Gamma(y) = \gamma \qquad \gamma \approx \beta}{\Gamma, x\colon \alpha \vdash_{\_} y\colon \beta}$$

  and we have two cases to consider:

  - $x = y$
    Since $\Gamma \vdash_{\_} U\colon \alpha$ with $\alpha = \gamma$ and $\gamma \approx \beta$ then we conclude $\Gamma \vdash_{\_} U\colon \beta$.
  - $x \neq y$
    The value is not affected by the substitution, then by rule TV-ID $\Gamma \vdash_{\_} y\colon \beta$ holds.

- Case V is $(\vec{y})P$
  By hypothesis

  $$\frac{\Gamma, x\colon \alpha \vdash_r P \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r (\vec{y})P\colon \mathsf{abs}(\vec{\beta})}$$

  Since $\Gamma \vdash_{\_} U\colon \alpha$, we apply the clause for Processes to $\Gamma, x\colon \alpha \vdash_r P$ and obtain $\Gamma \vdash_{r[U/x]} P[U/x]$. The names in $\vec{y}$ are bound, so they are not affected by the substitution, then by rule TV-ABS and by definition of substitution, we get $\Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} ((\vec{y})P)[U/x]\colon \mathsf{abs}(\vec{\beta})$ which concludes the case.

Substitution on Methods. By hypothesis, the following derivation tree holds.

$$\frac{\dfrac{\Gamma, x\colon \alpha \vdash_r P_1 \quad \Gamma, x\colon \alpha \vdash_\_ \vec{y_1}\colon \vec{\beta_1}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_1}\} \vdash_r (\vec{y_1})P_1\colon \mathsf{abs}(\vec{\beta_1})} \quad \ldots \quad \dfrac{\Gamma, x\colon \alpha \vdash_r P_n \quad \Gamma, x\colon \alpha \vdash_\_ \vec{y_n}\colon \vec{\beta_n}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_n}\} \vdash_r (\vec{y_n})P_n\colon \mathsf{abs}(\vec{\beta_n})}}{(\Gamma, x\colon \alpha) \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_r m\colon \rho}$$

By hypothesis $\Gamma \vdash_\_ U\colon \alpha$. We apply the clause for Processes to $\Gamma, x\colon \alpha \vdash_r P_1, \ldots, \Gamma, x\colon \alpha \vdash_r P_n$ and obtain $\Gamma \vdash_{r[U/x]} P_1[U/x], \ldots, \Gamma \vdash_{r[U/x]} P_n[U/x]$. Let $\delta_1 = \mathsf{abs}(\vec{\beta_1}), \ldots, \delta_n = \mathsf{abs}(\vec{\beta_n})$. Since the names in $\vec{y_1}, \ldots, \vec{y_n}$ are bound, they are not affected by substitution so $\Gamma \vdash_\_ \vec{y_1}\colon \vec{\beta_1}, \ldots, \Gamma \vdash_\_ \vec{y_n}\colon \vec{\beta_n}$ hold. By TM-METH rule, the following derivation tree holds

$$\frac{\dfrac{\Gamma \vdash_{r[U/x]} P_1[U/x] \quad \Gamma \vdash_\_ \vec{y_1}\colon \vec{\beta_1}}{\Gamma \setminus \{\vec{y_1}\} \vdash_{r[U/x]} (\vec{y_1})P_1[U/x]\colon \delta_1} \quad \ldots \quad \dfrac{\Gamma \vdash_{r[U/x]} P_n[U/x] \quad \Gamma \vdash_\_ \vec{y_n}\colon \vec{\beta_n}}{\Gamma \setminus \{\vec{y_n}\} \vdash_{r[U/x]} (\vec{y_n})P_n[U/x]\colon \delta_n}}{\Gamma \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_{r[U/x]} \{l_1 = (\vec{y_1})P_1[U/x] \ldots l_n = (\vec{y_n})P_n[U/x]\}\colon \rho}$$

where $\{l_1 = (\vec{y_1})P_1[U/x] \ldots l_n = (\vec{y_n})P_n[U/x]\}$ is by definition of substitution $\{l_1 = (\vec{y_1})P_1 \ldots l_n = (\vec{y_n})P_n\}[U/x]$.

Substitution on Processes. We proceed by case analysis on the structure of P, tracing the last typing rule applied.

- Case P is $\mathsf{inaction}$
  We derive by hypothesis, $\Gamma, x\colon \alpha \vdash_r \mathsf{inaction}$. Since $\Gamma \vdash_\_ U\colon \alpha$, by TP-INACT rule we conclude $\Gamma \vdash_{r[U/x]} \mathsf{inaction}$.

- Case P is $y\,!\,l[\vec{V}]$
  By hypothesis, the following derivation holds

  $$\frac{\Gamma, x\colon \alpha \vdash_r \vec{V}\colon \rho.l \quad \Gamma, x\colon \alpha \vdash_\_ y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma, x\colon \alpha \vdash_r y\,!\,l[\vec{V}]}$$

  Let $\vec{\beta} = \rho.l$, then the derivation of $\Gamma, x\colon \alpha \vdash_r \vec{V}\colon \vec{\beta}$ is

  $$\frac{\Gamma, x\colon \alpha \vdash_r V_1\colon \beta_1 \quad \ldots \quad \Gamma, x\colon \alpha \vdash_r V_n\colon \beta_n}{\Gamma, x\colon \alpha \vdash_r \vec{V}\colon \vec{\beta}}$$

  We have two sub-cases:

  - $x = y$
    By hypothesis $\Gamma \vdash_\_ U\colon \alpha$. We apply the clause for Values to $\Gamma, x\colon \alpha \vdash_r V_1\colon \beta_1, \ldots, \Gamma, x\colon \alpha \vdash_r V_n\colon \beta_n$ and conclude $\Gamma \vdash_{r[U/x]} \vec{V}[U/x]\colon \vec{\beta}$. Since $x = y$, we obtain $\Gamma \vdash_\_ U \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}$. By rule TP-MSG we conclude the sub-case.

    $$\frac{\Gamma \vdash_{r[U/x]} \vec{V}[U/x]\colon \rho.l \quad \Gamma \vdash_\_ U \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_{r[U/x]} U\,!\,l[\vec{V}[U/x]]}$$

    that can be written as $\Gamma \vdash_{r[U/x]} U\,!\,l[\vec{V}][U/x]$, by definition of substitution.

– $x \neq y$

By hypothesis $\Gamma \vdash\_ U\colon \alpha$. Then, we apply the clause for Values to $\Gamma, x\colon \alpha \vdash_r V_1\colon \beta_1, \ldots, \Gamma, x\colon \alpha \vdash_r V_n\colon \beta_n$, and get $\Gamma \vdash_{r[U/x]} \vec{V}[U/x]\colon \vec{\beta}$. Since $x \neq y$, $\Gamma \vdash\_ y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}$ holds and we conclude the sub-case by rule TP-MSG.

$$\frac{\Gamma \vdash_{r[U/x]} \vec{V}[U/x]\colon \rho.l \qquad \Gamma \vdash\_ y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_{r[U/x]} y\,!\,l[\vec{V}[U/x]]}$$

which is $\Gamma \vdash_{r[U/x]} y\,!\,l[\vec{V}][U/x]$, by definition of substitution.

- Case P is $y?m$ or P is $y?*m$
  These two cases hold in a very similar way. We illustrate the proof for the first case as an example. By hypothesis, $\Gamma, x\colon \alpha \vdash_r y?m$, so the following type derivation holds

$$\frac{\Gamma, x\colon \alpha \vdash_r m\colon \rho \qquad \Gamma, x\colon \alpha \vdash\_ y\colon \mathsf{ch}(\rho)}{\Gamma, x\colon \alpha \vdash_r y?m}$$

  We need to study two sub-cases:

  – Case $x = y$
  Since $\Gamma \vdash\_ U\colon \alpha$, we apply the clause for Methods to $\Gamma, x\colon \alpha \vdash_r m\colon \rho$ and get $\Gamma \vdash_{r[U/x]} m[U/x]\colon \rho$. By hypothesis, $\Gamma \vdash\_ y\colon \mathsf{ch}(\rho)$ so we get $\Gamma \vdash\_ U\colon \mathsf{ch}(\rho)$ and by rule TP-OBJ the following type derivation holds.

$$\frac{\Gamma \vdash_{r[U/x]} m[U/x]\colon \rho \qquad \Gamma \vdash\_ U\colon \mathsf{ch}(\rho)}{\Gamma \vdash_{r[U/x]} U?m[U/x]}$$

  that, by definition of substitution is written as $\Gamma \vdash_{r[U/x]} (U?m)[U/x]$.

  – $x \neq y$
  By hypothesis, $\Gamma \vdash\_ U\colon \alpha$, and applying the clause for Methods to $\Gamma, x\colon \alpha \vdash_r m\colon \rho$, we get $\Gamma \vdash_r m[U/x]\colon \rho$. Since $x \neq y$, y is not affected by substitution and by TP-OBJ rule the following derivation tree concludes the sub-case.

$$\frac{\Gamma \vdash_{r[U/x]} m[U/x]\colon \rho \qquad \Gamma \vdash\_ y\colon \mathsf{ch}(\rho)}{\Gamma \vdash_{r[U/x]} y?m[U/x]}$$

  that is written as $\Gamma \vdash_{r[U/x]} (y?m)[U/x]$, by definition of substitution.

- Case P is $\mathsf{out}[y, l[\vec{V}]]$
  By hypothesis,

$$\frac{\Gamma, x\colon \alpha \vdash_y (r\vec{V})\colon \rho.l \qquad \Gamma, x\colon \alpha \vdash\_ y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash_r \mathsf{out}[y, l[\vec{V}]]}$$

  where $\beta = \rho.l$ and the derivation of $\Gamma, x\colon \alpha \vdash_y (r\vec{V})\colon \vec{\beta}$ is

$$\frac{\Gamma, x\colon \alpha \vdash_y r\colon \beta_0 \quad \Gamma, x\colon \alpha \vdash_y V_1\colon \beta_1 \quad \ldots \quad \Gamma, x\colon \alpha \vdash_y V_n\colon \beta_n}{\Gamma, x\colon \alpha \vdash_y (r\vec{V})\colon \vec{\beta}}$$

We have four sub-cases to handle:

- Case $x = y$ and $x = r$
  Since by hypothesis, $\Gamma \vdash\_ U\colon \alpha$, we apply the clause for Values to $\Gamma, x\colon \alpha \vdash_y r\colon \beta_0$, $\Gamma, x\colon \alpha \vdash_y V_1\colon \beta_1$, …, $\Gamma, x\colon \alpha \vdash_y V_n\colon \beta_n$ and get $\Gamma \vdash_U U\colon \beta_0$, $\Gamma \vdash_U V_1[U/x]\colon \beta_1$, …, $\Gamma \vdash_U V_n[U/x]\colon \beta_n$. By hypothesis, $\Gamma \vdash\_ U\colon \alpha$ and $\Gamma, x\colon \alpha \vdash\_ y\colon \mathsf{dom}(\rho)$, so $\Gamma \vdash\_ U\colon \mathsf{dom}(\rho)$ holds. By rule TP-OUT we conclude the case

  $$\frac{\Gamma \vdash_U U\vec{V}[U/x]\colon \vec{\beta} \quad \Gamma \vdash\_ U\colon \mathsf{dom}(\rho)}{\Gamma \vdash_U \mathsf{out}[U, l[\vec{V}[U/x]]]}$$

  where

  $$\frac{\Gamma \vdash_U U\colon \beta_0 \quad \Gamma \vdash_U V_1[U/x]\colon \beta_1 \quad \ldots \quad \Gamma \vdash_U V_n[U/x]\colon \beta_n}{\Gamma \vdash_U U\vec{V}[U/x]\colon \vec{\beta}}$$

  that by definition of substitution is $\Gamma \vdash_U \mathsf{out}[U, l[\vec{V}]][U/x]$.
- Case $x \neq y$ and $x = r$
  Since by hypothesis, we have $\Gamma \vdash\_ U\colon \alpha$, applying the clause for Values to $\Gamma, x\colon \alpha \vdash_y r, \Gamma, x\colon \alpha \vdash_y V_1\colon \beta_1, \ldots, \Gamma, x\colon \alpha \vdash_y V_n\colon \beta_n$ we get $\Gamma \vdash_y U\colon \beta_0$, $\Gamma \vdash_y V_1[U/x]\colon \beta_1$, …, $\Gamma \vdash_y V_n[U/x]\colon \beta_n$. So, we have $\Gamma \vdash_y (U\vec{V})[U/x]\colon \vec{\beta}$. Name y is not affected by the substitution so $\Gamma \vdash\_ y\colon \mathsf{dom}(\rho)$ holds. By rule TP-OUT we obtain the following derivation tree

  $$\frac{\Gamma \vdash_y U\vec{V}[U/x]\colon \vec{\beta} \quad \Gamma \vdash\_ y\colon \mathsf{dom}(\rho)}{\Gamma \vdash_U \mathsf{out}[y, l[\vec{V}[U/x]]]}$$

  and the proof for $\Gamma \vdash_y U\vec{V}[U/x]\colon \vec{\beta}$ is

  $$\frac{\Gamma \vdash_y U\colon \beta_0 \quad \Gamma \vdash_y V_1[U/x]\colon \beta_1 \quad \ldots \quad \Gamma \vdash_y V_n[U/x]\colon \beta_n}{\Gamma \vdash_y U\vec{V}[U/x]\colon \vec{\beta}}$$

  that is written as $\Gamma \vdash_U \mathsf{out}[y, l[\vec{V}]][U/x]$, by definition of substitution.
- Case $x = y$ and $x \neq r$
  By hypothesis $\Gamma \vdash\_ U\colon \alpha$ and we apply the clause for Values to $\Gamma, x\colon \alpha \vdash_y r\colon \beta_0, \Gamma, x\colon \alpha \vdash_y V_1\colon \beta_1$, …, $\Gamma, x\colon \alpha \vdash_y V_n\colon \beta_n$ and obtain $\Gamma \vdash_U r\colon \beta_0, \Gamma \vdash_U V_1[U/x]\colon \beta_1$, …, $\Gamma \vdash_U V_n[U/x]\colon \beta_n$. Since $\Gamma \vdash\_ U\colon \alpha$ and $\Gamma, x\colon \alpha \vdash\_ y\colon \mathsf{dom}(\rho)$, then $\Gamma \vdash\_ U\colon \mathsf{dom}(\rho)$. By rule TP-OUT the following derivation holds

  $$\frac{\Gamma \vdash_U r\vec{V}[U/x]\colon \vec{\beta} \quad \Gamma \vdash\_ U\colon \mathsf{dom}(\rho)}{\Gamma \vdash_r \mathsf{out}[U, l[\vec{V}[U/x]]]}$$

where

$$\frac{\Gamma \vdash_U r\colon \beta_0 \quad \Gamma \vdash_U V_1[U/x]\colon \beta_1 \quad \ldots \quad \Gamma \vdash_U V_n[U/x]\colon \beta_n}{\Gamma \vdash_U r\vec{V}[U/x]\colon \vec{\beta}}$$

that is written as $\Gamma \vdash_r \mathsf{out}[U, l[\vec{V}]][U/x]$, by definition of substitution.

– Case $x \neq y$ and $x \neq r$
By hypothesis and the clause for Values the following derivation tree concludes the case

$$\frac{\Gamma \vdash_y r\vec{V}[U/x]\colon \vec{\beta} \quad \Gamma \vdash_\_ y\colon \mathsf{dom}(\rho)}{\Gamma \vdash_r \mathsf{out}[y, l[\vec{V}[U/x]]]}$$

where

$$\frac{\Gamma \vdash_y r\colon \beta_0 \quad \Gamma \vdash_y V_1[U/x]\colon \beta_1 \quad \ldots \quad \Gamma \vdash_y V_n[U/x]\colon \beta_n}{\Gamma \vdash_y r\vec{V}[U/x]\colon \vec{\beta}}$$

that is written as $\Gamma \vdash_r \mathsf{out}[y, l[\vec{V}]][U/x]$, by definition of substitution.

- Case P is $((\vec{y})Q)\vec{V}$
By hypothesis,

$$\frac{\dfrac{\Gamma, x\colon \alpha \vdash_r Q \quad \Gamma, x\colon \alpha \vdash_\_ \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r (\vec{y})Q\colon \mathsf{abs}(\vec{\beta})} \quad \dfrac{\begin{array}{c}(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r V_1\colon \beta_1 \\ \vdots \\ (\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r V_n\colon \beta_n\end{array}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r \vec{V}\colon \vec{\beta}}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r ((\vec{y})Q)\vec{V}}$$

Names in $\vec{y}$ are bound, so they are not affected by the substitution and $\Gamma \vdash_\_ \vec{y}\colon \vec{\beta}$ holds. Since $\Gamma \vdash_\_ U\colon \alpha$, we apply induction hypothesis to $\Gamma, x\colon \alpha \vdash_r Q$, clause for Values to $(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r V_1\colon \beta_1, \ldots, (\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_r V_n\colon \beta_n$, and by rule TP-APP we derive the following tree

$$\frac{\dfrac{\Gamma \vdash_{r[U/x]} Q[U/x] \quad \Gamma \vdash_\_ \vec{y}\colon \vec{\beta}}{\Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} (\vec{y})Q[U/x]\colon \mathsf{abs}(\vec{\beta})} \quad \dfrac{\begin{array}{c}\Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} V_1[U/x]\colon \beta_1 \\ \vdots \\ \Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} V_n[U/x]\colon \beta_n\end{array}}{\Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} \vec{V}[U/x]\colon \vec{\beta}}}{\Gamma \setminus \{\vec{y}\} \vdash_{r[U/x]} ((\vec{y})Q[U/x])\vec{V}[U/x]}$$

that is , by definition of substitution, the same as $\Gamma \vdash_{r[U/x]} ((\vec{y})Q\vec{V})[U/x]$.

- Case P is $P_1 \mid P_2$, P is $\mathsf{new}\ y\ Q$, or P is $\mathsf{in}[Q]$.
These cases hold trivially. We illustrate the proof for the first case as an example. By hypothesis, the following type derivation holds

$$\frac{\Gamma, x\colon \alpha \vdash_r P_1 \quad \Gamma, x\colon \alpha \vdash_r P_2}{\Gamma, x\colon \alpha \vdash_r P_1 \mid P_2}$$

by induction hypothesis applied to $\Gamma, x\colon \alpha \vdash_r P_1$ and $\Gamma, x\colon \alpha \vdash_r P_2$, and by TP-PAR we conclude

$$\frac{\Gamma \vdash_{r[U/x]} P_1[U/x] \qquad \Gamma \vdash_{r[U/x]} P_2[U/x]}{\Gamma \vdash_{r[U/x]} P_1[U/x] \mid P_2[U/x]}$$

that is, by definition of substitution, $\Gamma \vdash_{r[U/x]} (P_1 \mid P_2)[U/x]$.

- Case $\Gamma, x\colon \alpha \vdash_r \mathsf{mkdom}[y, m\rangle S, P]\ \mathsf{in}\ R$
  The following derivation tree holds

$$\frac{\dfrac{\begin{array}{c}\Gamma, x\colon \alpha \vdash_y m\colon \rho \\ \Gamma, x\colon \alpha \vdash_y S \mid P \\ \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)\end{array}}{\Gamma, x\colon \alpha \vdash y\{m\rangle S\}[P]} \qquad \Gamma, x\colon \alpha \vdash_r R}{(\Gamma, x\colon \alpha) \setminus \{y\} \vdash_r \mathsf{mkdom}[y, m\rangle S, P]\ \mathsf{in}\ R}$$

  where

$$\frac{\Gamma, x\colon \alpha \vdash_y S \qquad \Gamma, x\colon \alpha \vdash_y P}{\Gamma, x\colon \alpha \vdash_y S \mid P}$$

  We apply induction hypothesis to $\Gamma, x\colon \alpha \vdash_y P$, to $\Gamma, x\colon \alpha \vdash_y S$, and to $\Gamma, x\colon \alpha \vdash_r R$, and obtain $\Gamma \vdash_y P[U/x]$, $\Gamma \vdash_y S[U/x]$, and $\Gamma \vdash_{r[U/x]} R[U/x]$, respectively. By the clause for Methods applied to $\Gamma, x\colon \alpha \vdash_y m\colon \rho$ we get $\Gamma \vdash_y m[U/x]\colon \rho$ and since y is bound, $\Gamma \vdash_{\_} y\colon \mathsf{dom}(\rho)$ holds. Therefore, by TP-MKD rule we conclude the following derivation tree

$$\frac{\dfrac{\begin{array}{c}\Gamma \vdash_y m[U/x]\colon \rho \\ \Gamma \vdash_y S[U/x] \mid P[U/x] \\ \Gamma \vdash_{\_} y\colon \mathsf{dom}(\rho)\end{array}}{\Gamma \vdash y\{m[U/x]\rangle S[U/x]\}[P[U/x]]} \qquad \Gamma \vdash_{r[U/x]} R[U/x]}{\Gamma \setminus \{y\} \vdash_{r[U/x]} \mathsf{mkdom}[y, m[U/x]\rangle S[U/x], P[U/x]]\ \mathsf{in}\ R[U/x]}$$

  where

$$\frac{\Gamma \vdash_y S[U/x] \quad \Gamma \vdash_y P[U/x]}{\Gamma \vdash_y S[U/x] \mid P[U/x]}$$

  that is $\Gamma \vdash_{r[U/x]} (\mathsf{mkdom}[y, m\rangle S, P]\ \mathsf{in}\ R)[U/x]$, by definition of substitution.

Substitution on Networks. The proof for the second case is also obtained by straightforward induction on the typing of $\Gamma \vdash N$, and analyzing the structure of N. We present only the case that is different from the first clause.

- Case N is $y\{m\rangle S\}[P]$
  By hypothesis

$$\frac{\Gamma, x\colon \alpha \vdash_y m\colon \rho \qquad \dfrac{\begin{array}{c}\Gamma, x\colon \alpha \vdash_y S \\ \Gamma, x\colon \alpha \vdash_y P\end{array}}{\Gamma, x\colon \alpha \vdash_y S \mid P} \qquad \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash y\{m\rangle S\}[P]}$$

  We have two sub-cases:

– $x = y$
From $\Gamma, x\colon \alpha \vdash_x m\colon \rho$ and $\Gamma \vdash_\_ U\colon \alpha$ we apply the clause for Methods and obtain $\Gamma \vdash_U m[U/x]\colon \rho$. By induction hypothesis applied to $\Gamma, x\colon \alpha \vdash_y P$ and $\Gamma, x\colon \alpha \vdash_y S$, we get, respectively, $\Gamma \vdash_U P[U/x]$ and $\Gamma \vdash_U S[U/x]$. From $\Gamma \vdash_\_ y\colon \mathsf{dom}(\rho)$ and by hypothesis, we get $\Gamma \vdash_\_ U\colon \mathsf{dom}(\rho)$. By TN-DOM rule the following derivation tree holds

$$\frac{\Gamma \vdash_U m[U/x]\colon \rho \qquad \dfrac{\begin{array}{c}\Gamma \vdash_U S[U/x] \\ \Gamma \vdash_U P[U/x]\end{array}}{\Gamma \vdash_U S[U/x] \mid P[U/x]} \qquad \Gamma \vdash_\_ U\colon \mathsf{dom}(\rho)}{\Gamma \vdash U\{m[U/x] \between S[t/s]\}[P[t/s]]}$$

that is $\Gamma \vdash (U\{m \between S\}[P])[U/x]$, by definition of substitution.

– $x \neq y$
Since $\Gamma, x\colon \alpha \vdash_y m\colon \rho$ and $\Gamma \vdash_\_ U\colon \alpha$, applying the clause for Methods we get $\Gamma \vdash_y m[U/x]\colon \rho$. By induction hypothesis applied to $\Gamma, x\colon \alpha \vdash_y P$ and $\Gamma, x\colon \alpha \vdash_y S$, we get, respectively, $\Gamma \vdash_y P[U/x]$ and $\Gamma \vdash_y S[U/x]$. Since y is not affected by the substitution, $\Gamma \vdash_\_ y\colon \mathsf{dom}(\rho)$ holds and the following derivation tree concludes the case by TN-DOM rule.

$$\frac{\Gamma \vdash_y m[t/s]\colon \rho \qquad \dfrac{\begin{array}{c}\Gamma \vdash_y S[U/x] \\ \Gamma \vdash_y P[U/x]\end{array}}{\Gamma \vdash_y S[U/x] \mid P[U/x]} \qquad \Gamma \vdash_\_ y\colon \mathsf{dom}(\rho)}{\Gamma \vdash y\{m[U/x] \between S[U/x]\}[P[U/x]]}$$

that may be rewritten as $\Gamma \vdash (y\{m \between S\}[P])[U/x]$, by definition of substitution.

$\diamond$

Lemma A.2 (Strengthening)

1. *If* $x \notin \mathrm{fn}(V) \cup \{s\}$ *and* $\Gamma, x\colon \alpha \vdash_s V\colon \beta$, *then* $\Gamma \vdash_s V\colon \beta$.
2. *If* $\Gamma, x\colon \alpha \vdash_s m\colon \rho$ *and* $x \notin \mathrm{fn}(m) \cup \{s\}$, *then* $\Gamma \vdash_s m\colon \rho$.
3. *If* $\Gamma, x\colon \alpha \vdash_s P$ *and* $x \notin \mathrm{fn}(P) \cup \{s\}$, *then* $\Gamma \vdash_s P$.
4. *If* $\Gamma, x\colon \alpha \vdash N$ *and* $x \notin \mathrm{fn}(N)$, *then* $\Gamma \vdash N$.

Proof. The proof of the first three results is by mutual induction on the typing derivation.

Strengthening for Values.

- Case V is y
Since $x \neq y$, this case holds by hypothesis and we conclude $\Gamma \vdash_\_ y\colon \beta$.

- Case V is $(\vec{y})P$

$$\frac{\Gamma, x\colon \alpha \vdash_s P \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s (\vec{y})P\colon \mathsf{abs}(\vec{\beta})}$$

Since $x \notin \mathrm{fn}(P) \cup \{s\}$, by the clause for Processes applied to $\Gamma, x\colon \alpha \vdash_s P$, we get $\Gamma \vdash_s P$. Names in $\vec{y}$ are bound so $\Gamma \vdash_{\_} \vec{y}\colon \vec{\beta}$ holds and by rule TV-ABS we conclude

$$\frac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} \vec{y}\colon \vec{\beta}}{\Gamma \setminus \{\vec{y}\} \vdash_s (\vec{y})P\colon \mathsf{abs}(\vec{\beta})}$$

Strengthening for Methods. By straightforward induction on the derivation tree of $(\Gamma, x\colon \alpha) \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_r m\colon \rho$.

By hypothesis,

$$\frac{\dfrac{\Gamma, x\colon \alpha \vdash_s P_1 \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y_1}\colon \vec{\beta_1}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_1}\} \vdash_s (\vec{y_1})P_1\colon \mathsf{abs}(\vec{\beta_1})} \qquad \dfrac{\Gamma, x\colon \alpha \vdash_s P_n \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y_n}\colon \vec{\beta_n}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_n}\} \vdash_s (\vec{y_n})P_n\colon \mathsf{abs}(\vec{\beta_n})}}{(\Gamma, x\colon \alpha) \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_s \{l_1 = (\vec{y_1})P_1 \ldots l_n = (\vec{y_n})P_n\}\colon \rho}$$

Since $x \notin \mathrm{fn}(m) \cup \{s\}$, we conclude $\Gamma \vdash_{\_} \vec{y_1}\colon \vec{\beta_1}, \ldots, \Gamma \vdash_{\_} \vec{y_n}\colon \vec{\beta_n}$ and applying the clause for Processes to $\Gamma, x\colon \alpha \vdash_s P_1, \ldots, \Gamma, x\colon \alpha \vdash_s P_n$, the following derivation tree holds.

$$\frac{\dfrac{\Gamma \vdash_s P_1 \qquad \Gamma \vdash_{\_} \vec{y_1}\colon \vec{\beta_1}}{\Gamma \setminus \{\vec{y_1}\} \vdash_s (\vec{y_1})P_1\colon \mathsf{abs}(\vec{\beta_1})} \qquad \dfrac{\Gamma \vdash_s P_n \qquad \Gamma \vdash_{\_} \vec{y_n}\colon \vec{\beta_n}}{\Gamma \setminus \{\vec{y_n}\} \vdash_s (\vec{y_n})P_n\colon \mathsf{abs}(\vec{\beta_n})}}{\Gamma \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_s \{l_1 = (\vec{y_1})P_1 \ldots l_n = (\vec{y_n})P_n\}\colon \rho}$$

Strengthening for Processes. Proved by induction of the derivation tree of $\Gamma \vdash_s P$. We proceed by case analysis on the structure of P, analyzing the last typing rule applied.

- Case P is $\mathsf{inaction}$
  By hypothesis, $\Gamma, x\colon \alpha \vdash_s \mathsf{inaction}$, then $\Gamma \vdash_s \mathsf{inaction}$.

- Case P is $y\,!\,l[\vec{V}]$
  Let $\vec{\beta} = \rho.l$. The following derivation tree holds

$$\frac{\dfrac{\begin{array}{l}\Gamma, x\colon \alpha \vdash_s V_1\colon \beta_1 \\ \vdots \\ \Gamma, x\colon \alpha \vdash_s V_n\colon \beta_n\end{array}}{\Gamma, x\colon \alpha \vdash_s \vec{V}\colon \vec{\beta}} \qquad \Gamma, x\colon \alpha \vdash_{\_} y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma, x\colon \alpha \vdash_s y\,!\,l[\vec{V}]}$$

Since $x \notin \mathrm{fn}(\vec{V}) \cup \{s\}$, we apply the clause for Values and rule TV-SEQ to $\Gamma, x\colon \alpha \vdash_s V_1, \ldots \Gamma, x\colon \alpha \vdash_s V_n$, and obtain $\Gamma \vdash_s \vec{V}\colon \vec{\beta}$. By hypothesis, $x \neq y$, so $\Gamma \vdash_{\_} y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}$ holds. Applying TP-MSG rule we conclude the case.

- Case P is $y?m$ or P is $y?*m$
  These two cases are proved similarly. We prove only the first case. By hypothesis,
  $$\frac{\Gamma, x\colon \alpha \vdash_s m\colon \rho \qquad \Gamma \vdash_\_ y\colon \mathsf{ch}(\rho)}{\Gamma, x\colon \alpha \vdash_s y?m}$$
  Since, $x \notin \mathrm{fn}(m) \cup \{s\}$, by the clause for Methods applied to $\Gamma, x\colon \alpha \vdash_s m\colon \rho$ we get $\Gamma \vdash_s m\colon \rho$. By hypothesis, $x \neq y$, so $\Gamma \vdash_\_ y\colon \mathsf{ch}(\rho)$ holds. Therefore, applying TP-OBJ rule we conclude the case
  $$\frac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_\_ y\colon \mathsf{ch}(\rho)}{\Gamma \vdash_s y?m}$$

- Case $\mathsf{out}[y, l[\vec{V}]]$
  Let $\vec{\beta} = \rho.l$.
  $$\frac{\begin{array}{c}\Gamma, x\colon \alpha \vdash_s s\colon \beta_0 \\ \Gamma, x\colon \alpha \vdash_s V_1\colon \beta_1 \\ \vdots \\ \Gamma, x\colon \alpha \vdash_s V_n\colon \beta_n\end{array}}{\Gamma, x\colon \alpha \vdash_s (s\vec{V})\colon \vec{\beta}} \qquad \Gamma, x\colon \alpha \vdash_\_ y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash_s \mathsf{out}[y, l[\vec{V}]]}$$
  Since $x \notin \{s\} \cup \mathrm{fn}(V_i), i \in I$, by the clause for values we get $\Gamma \vdash_s (s\vec{V})\colon \vec{\beta}$ and by TP-OUT rule, we conclude the case.

- Case P is $((\vec{y})Q)\vec{V}$
  By hypothesis,
  $$\frac{\dfrac{\Gamma, x\colon \alpha \vdash_s Q \qquad \Gamma, x\colon \alpha \vdash_\_ \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s (\vec{y})Q\colon \mathsf{abs}(\vec{\beta})} \qquad (\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s ((\vec{y})Q)\vec{V}}$$
  Since $x \notin (\{s\} \cup \mathrm{fn}(Q) \cup \mathrm{fn}(\vec{V})) \setminus \{\vec{y}\}$, we apply induction hypothesis to $\Gamma, x\colon \alpha \vdash_s Q$, and the clause for Values to $(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}$ and deduce, respectively, $\Gamma \vdash_s Q$ and $\Gamma \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}$. Names in $\vec{y}$ are bound, so $\Gamma \vdash_\_ \vec{y}\colon \vec{\beta}$ holds. Therefore, we conclude $\Gamma \vdash_s ((\vec{y})Q)\vec{V}$ by rule TP-APP.

- Case P is $P_1 \mid P_2$, P is $\mathsf{new}\ y\ Q$ or P is $\mathsf{in}[Q]$
  These cases hold trivially. We prove only the first case. By hypothesis,
  $$\frac{\Gamma, x\colon \alpha \vdash_s P_1 \qquad \Gamma, x\colon \alpha \vdash_s P_2}{\Gamma, x\colon \alpha \vdash_s P_1 \mid P_2}$$
  Since by hypothesis, $x \notin \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2) \cup \{s\}$, we apply induction hypothesis to $\Gamma, x\colon \alpha \vdash_s P_1$ and $\Gamma, x\colon \alpha \vdash_s P_2$, and get $\Gamma \vdash_s P_1$ and $\Gamma \vdash_s P_2$. By rule TP-PAR, we conclude $\Gamma \vdash_s P_1 \mid P_2$.

- Case $\mathsf{mkdom}[y, m\rangle S, P]$ in R

$$\dfrac{\dfrac{\Gamma, x\colon \alpha \vdash_y m\colon \rho \quad \Gamma, x\colon \alpha \vdash_y S \mid P \quad \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash_s y\{m\rangle S\}[P]} \quad \Gamma, x\colon \alpha \vdash_s R}{(\Gamma, x\colon \alpha) \setminus y \vdash_s \mathsf{mkdom}[y, m\rangle S, P] \text{ in } R}$$

  where

$$\dfrac{\Gamma, x\colon \alpha \vdash_y S \quad \Gamma, x\colon \alpha \vdash_y P}{\Gamma, x\colon \alpha \vdash_y S \mid P}$$

  Since $x \notin (\mathrm{fn}(m) \cup \mathrm{fn}(S) \cup \mathrm{fn}(P) \cup \mathrm{fn}(R) \cup \{s\}) \setminus \{y\}$, we apply induction hypothesis to $\Gamma, x\colon \alpha \vdash_s S$, $\Gamma, x\colon \alpha \vdash_s P$, and $\Gamma, x\colon \alpha \vdash_s R$, and the clause for methods to $\Gamma, x\colon \alpha \vdash_s m$. Name y is bound, so $\Gamma \vdash_{\_} y\colon \mathsf{dom}(\rho)$ holds, thus, by rule TP-MKD, we conclude the case.

Strengthening for Networks. The proof for the second case is obtained by straightforward induction on the typing of $\Gamma \vdash N$. We proceed by case analysis on the structure of N, analyzing the last typing rule applied. We present the case that is different from the first clause.

- Case N is $y\{m\rangle S\}[P]$
  By hypothesis,

$$\dfrac{\Gamma, x\colon \alpha \vdash_y m\colon \rho \quad \dfrac{\Gamma, x\colon \alpha \vdash_y S \quad \Gamma, x\colon \alpha \vdash_y P}{\Gamma, x\colon \alpha \vdash_y S \mid P} \quad \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash y\{m\rangle S\}[P]}$$

  Since $x \notin \{y\} \cup \mathrm{fn}(m) \cup \mathrm{fn}(S) \cup \mathrm{fn}(P)$, $\Gamma \vdash_{\_} y\colon \mathsf{dom}(\rho)$ holds and we apply the clause for Methods to $\Gamma, x\colon \alpha \vdash_s m$ and clause 1 of this lemma to $\Gamma, x\colon \alpha \vdash_s S$, and $\Gamma, x\colon \alpha \vdash_s P$. Thus, we deduce, respectively, $\Gamma \vdash_y m\colon \rho$, $\Gamma \vdash_y S$ and $\Gamma \vdash_y P$. By rule TN-DOM, we conclude $\Gamma \vdash y\{m\rangle S\}[P]$.

◇

Lemma A.3 (Weakening)

1. *If* $\Gamma \vdash_s V\colon \beta$, *then* $\Gamma, x\colon \alpha \vdash_s V\colon \beta$.
2. *If* $\Gamma \vdash_s m\colon \rho$, *then* $\Gamma, x\colon \alpha \vdash_s m\colon \rho$.
3. *If* $\Gamma \vdash_s P$, *then* $\Gamma, x\colon \alpha \vdash_s P$.
4. *If* $\Gamma \vdash N$, *then* $\Gamma, x\colon \alpha \vdash N$.

Proof. The proof of the first three results is by mutual induction on the typing derivation.

Weakening for Values Follows by straightforward induction of the structure of V, analyzing the last typing rule.

- Case V is y
  By hypothesis,
  $$\frac{\Gamma(y) = \gamma \qquad \gamma \approx \beta}{\Gamma \vdash_{\_} y\colon \beta}$$
  Since x not in Γ, then $\Gamma, x\colon \alpha \vdash_s y\colon \beta$ holds.

- Case V is $(\vec{y})P$
  By hypothesis,
  $$\frac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} \vec{y}\colon \vec{\beta}}{\Gamma \setminus \{\vec{y}\} \vdash_s (\vec{y})P\colon \mathsf{abs}(\vec{\beta})}$$
  Name x is not defined in Γ, so $\Gamma, x\colon \alpha \vdash_{\_} \vec{y}\colon \vec{\beta}$ holds. Applying the clause for Processes to $\Gamma \vdash_s P$, we obtain $\Gamma, x\colon \alpha \vdash_s P$ and rule TV-ABS concludes the case
  $$\frac{\Gamma, x\colon \alpha \vdash_s P \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s (\vec{y})P\colon \mathsf{abs}(\vec{\beta})}$$

Weakening for Methods. By hypothesis, the following derivation holds

$$\frac{\dfrac{\Gamma \vdash_s P_1}{\Gamma \setminus \{\vec{y_1}\} \vdash_s (\vec{y_1})P_1\colon \mathsf{abs}(\vec{\beta_1})} \quad \ldots \quad \dfrac{\Gamma \vdash_s P_n}{\Gamma \setminus \{\vec{y_n}\} \vdash_s (\vec{y_n})P_n\colon \mathsf{abs}(\vec{\beta_n})}}{\Gamma \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_s m\colon \rho}$$

Since x not in Γ, we apply the clause for Processes to $\Gamma \vdash_s P_1, \ldots, \Gamma \vdash_s P_n$ and the following derivation holds

$$\frac{\dfrac{\Gamma, x\colon \alpha \vdash_s P_1}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_1}\} \vdash_s (\vec{y})P_1\colon \mathsf{abs}(\vec{\beta_1})} \quad \ldots \quad \dfrac{\Gamma, x\colon \alpha \vdash_s P_n}{(\Gamma, x\colon \alpha) \setminus \{\vec{y_n}\} \vdash_s (\vec{x_n})P_n\colon \mathsf{abs}(\vec{\beta_n})}}{(\Gamma, x\colon \alpha) \setminus (\{\vec{y_1}\} \cup \ldots \cup \{\vec{y_n}\}) \vdash_s m\colon \rho}$$

Weakening for Processes.

- Case P is inaction
  If $\Gamma \vdash_s$ inaction, then $\Gamma, x\colon \alpha \vdash_s$ inaction.

- Case P is $y\,!\,l[\vec{V}]$
  Let $\vec{\beta} = \rho.l$. By rule TP-MSG, we obtain
  $$\frac{\dfrac{\Gamma \vdash_s V_1\colon \beta_1 \quad \ldots \quad \Gamma \vdash_s V_n\colon \beta_n}{\Gamma \vdash_s \vec{V}\colon \vec{\beta}} \qquad \Gamma \vdash_{\_} y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_s y\,!\,l[\vec{V}]}$$
  By hypothesis, x is not defined in Γ, so $\Gamma, x\colon \alpha \vdash_{\_} y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}$ holds and we apply the clause for values to $\Gamma \vdash_s V_1, \ldots, \Gamma \vdash_s V_n$, getting the following derivation tree

$$\frac{\Gamma, x\colon \alpha \vdash_s \vec{V}\colon \vec{\beta} \qquad \Gamma, x\colon \alpha \vdash_{\_} y \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma, x\colon \alpha \vdash_s y\,!\,l[\vec{V}]}$$

and the proof for $\Gamma, x\colon \alpha \vdash_s \vec{V}\colon \beta$ is

$$\frac{\Gamma, x\colon \alpha \vdash_s V_1\colon \beta_1 \qquad \ldots \qquad \Gamma, x\colon \alpha \vdash_s V_n\colon \beta_n}{\Gamma, x\colon \alpha \vdash_s \vec{V}\colon \vec{\beta}}$$

which concludes the case.

- Case P is $y?m$ or P is $y?*m$
  These two cases hold similarly. We illustrate the proof for the first case as an example.

  By hypothesis, we obtain

  $$\frac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_{\_} y\colon \mathsf{ch}(\rho)}{\Gamma \vdash_s y?m}$$

  Since x is not defined in Γ, then $\Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{ch}(\rho)$ holds. Applying the clause for methods to $\Gamma \vdash_s m$, we get

  $$\frac{\Gamma, x\colon \alpha \vdash_s m\colon \rho \qquad \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{ch}(\rho)}{\Gamma, x\colon \alpha \vdash_s y?m}$$

  that concludes the case.

- Case P is $\mathsf{out}[r, l[\vec{V}]]$
  Let $\vec{\beta} = \rho.l$.

  $$\frac{\dfrac{\Gamma \vdash_r s\colon \beta_0 \qquad \Gamma \vdash_r V_1\colon \beta_1 \ \ldots\ \Gamma \vdash_r V_n\colon \beta_n}{\Gamma \vdash_r (s\vec{V})\colon \vec{\beta}} \qquad \Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho)}{\Gamma \vdash_s \mathsf{out}[r, l[\vec{V}]]}$$

  By hypothesis, x is not in Γ, so $\Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho)$ holds. Applying the clause for Values to $\Gamma \vdash_r s\colon \beta_0$, $\Gamma \vdash_r V_1\colon \beta_1, \ldots, \Gamma \vdash_r V_n\colon \beta_n$, we get $\Gamma, x\colon \alpha \vdash_r (s\vec{V})\colon \vec{\beta}$. By rule TP-OUT we conclude the case:

  $$\frac{\Gamma, x\colon \alpha \vdash_r (s\vec{V})\colon \vec{\beta} \qquad \Gamma, x\colon \alpha \vdash_{\_} r\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash_s \mathsf{out}[r, l[\vec{V}]]}$$

- Case P is $((\vec{y})Q)\vec{V}$
  By hypothesis

  $$\frac{\dfrac{\Gamma \vdash_s Q \qquad \Gamma \vdash_{\_} \vec{y}\colon \vec{\beta}}{\Gamma \setminus \{\vec{y}\} \vdash_s (\vec{y})Q\colon \mathsf{abs}(\vec{\beta})} \qquad \dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s V_1\colon \beta_1 \ldots \Gamma \setminus \{\vec{y}\} \vdash_s V_n\colon \beta_n}{\Gamma \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}}}{\Gamma \setminus \{\vec{y}\} \vdash_s ((\vec{y})Q)\vec{V}}$$

  Since, by hypothesis, x is not defined in Γ, then $\Gamma, x\colon \alpha \vdash_{\_} y\colon \beta$ holds. Applying induction hypothesis to $\Gamma \vdash_s Q$ and the clause for Values to $\Gamma \setminus \{\vec{y}\} \vdash_s V_1\colon \beta_1, \ldots, \Gamma \setminus \{\vec{y}\} \vdash_s V_n\colon \beta_n$, the following derivation holds

$$\dfrac{\dfrac{\Gamma, x\colon \alpha \vdash_s Q \qquad \Gamma, x\colon \alpha \vdash_{\_} \vec{y}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s (\vec{y})Q\colon \mathsf{abs}(\vec{\beta})} \qquad (\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s ((\vec{y})Q)\vec{V}}$$

where

$$\dfrac{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s V_1\colon \beta_1 \qquad \ldots \qquad (\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s V_n\colon \beta_n}{(\Gamma, x\colon \alpha) \setminus \{\vec{y}\} \vdash_s \vec{V}\colon \vec{\beta}}$$

- Case P is $P_1 \mid P_2$, P is $\mathsf{new}\ y\ Q$, or P is $\mathsf{in}[Q]$
  These cases hold trivially. We illustrate the proof for the first case as an example. By hypothesis, the following derivation holds

  $$\dfrac{\Gamma \vdash_s P_1 \qquad \Gamma \vdash_s P_2}{\Gamma \vdash_s P_1 \mid P_2}$$

  Since name x is not defined in Γ, we apply induction hypothesis to $\Gamma \vdash_s P_1$ and $\Gamma \vdash_s P_2$, and by rule TP-PAR the following derivation concludes the case.

  $$\dfrac{\Gamma, x\colon \alpha \vdash_s P_1 \qquad \Gamma, x\colon \alpha \vdash_s P_2}{\Gamma, x\colon \alpha \vdash_s P_1 \mid P_2}$$

- Case P is $\mathsf{mkdom}[r, m\rangle S, Q]\ \mathsf{in}\ R$

  $$\dfrac{\dfrac{\Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho) \qquad \dfrac{\Gamma \vdash_r Q \qquad \Gamma \vdash_r S}{\Gamma \vdash_r S \mid Q} \qquad \Gamma \vdash_r m\colon \rho}{\Gamma \vdash r\{m\rangle S\}[Q]} \qquad \Gamma \vdash_s R}{\Gamma \setminus \{r\} \vdash_s \mathsf{mkdom}[r, m\rangle S, Q]\ \mathsf{in}\ R}$$

  By hypothesis, x is not in Γ, so we apply the clause for Methods to $\Gamma \vdash_r m\colon \rho$ and induction hypothesis to $\Gamma \vdash_r S$, $\Gamma \vdash_r Q$, and $\Gamma \vdash_s R$. By rule TP-MKD the following derivation holds

  $$\dfrac{\dfrac{\begin{array}{c}\Gamma, x\colon \alpha \vdash_r S \mid Q \\ \Gamma, x\colon \alpha \vdash_r m\colon \rho \\ \Gamma, x\colon \alpha \vdash_{\_} r\colon \mathsf{dom}(\rho)\end{array}}{\Gamma, x\colon \alpha \vdash r\{m\rangle S\}[Q]} \qquad \Gamma, x\colon \alpha \vdash_s R}{(\Gamma, x\colon \alpha) \setminus \{r\} \vdash_s \mathsf{mkdom}[r, m\rangle S, Q]\ \mathsf{in}\ R}$$

  where

  $$\dfrac{\Gamma, x\colon \alpha \vdash_r Q \qquad \Gamma, x\colon \alpha \vdash_r S}{\Gamma, x\colon \alpha \vdash_r S \mid Q}$$

Weakening for Networks The proof is obtained by a straightforward induction on the typing tree of $\Gamma \vdash N$. We proceed by case analysis on the structure of N, analyzing the last typing rule applied. We present the case that is different from the first clause.

- Case N is $y\{m\!\rangle S\}[P]$
  By hypothesis,

$$\dfrac{\Gamma \vdash_y m\colon \rho \qquad \dfrac{\Gamma \vdash_y S \qquad \Gamma \vdash_y P}{\Gamma \vdash_y S \mid P} \qquad \Gamma \vdash_{\_} y\colon \mathsf{dom}(\rho)}{\Gamma \vdash y\{m\!\rangle S\}[P]}$$

  The sequent $\Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)$ holds, since x is not defined in Γ. Applying the clause for Methods to $\Gamma \vdash_y m\colon \rho$, we obtain $\Gamma, x\colon \alpha \vdash_y m$. The following derivation tree holds by clause 1 of the current lemma applied to $\Gamma \vdash_y S$ and $\Gamma \vdash_y P$, and by rule TN-DOM.

$$\dfrac{\Gamma, x\colon \alpha \vdash_y m\colon \rho \qquad \dfrac{\dfrac{\Gamma, x\colon \alpha \vdash_y S}{\Gamma, x\colon \alpha \vdash_y P}}{\Gamma, x\colon \alpha \vdash_y S \mid P} \qquad \Gamma, x\colon \alpha \vdash_{\_} y\colon \mathsf{dom}(\rho)}{\Gamma, x\colon \alpha \vdash y\{m\!\rangle S\}[P]}$$

$\diamond$

Lemma A.4 (Congruence Lemma)

1. *If* $\Gamma \vdash_s P$ *and* $P \equiv Q$*, then* $\Gamma \vdash_s Q$.
2. *If* $\Gamma \vdash N$ *and* $N \equiv L$*, then* $\Gamma \vdash N$.

Proof.

Congruence for Processes. By induction on the derivation tree of $\Gamma \vdash_s P$. We proceed by case analysis on each rule of the congruence relation defined in figure 2, inducting on the last rule applied.

- Case $\mathsf{new}\ x\ P \mid Q \equiv \mathsf{new}\ x\ (P \mid Q)$, if x not free in Q.

$$\dfrac{\dfrac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} x\colon \mathsf{ch}(\_)}{\Gamma \setminus \{x\} \vdash_s \mathsf{new}\ x\ P} \qquad \Gamma \setminus \{x\} \vdash_s Q}{\Gamma \setminus \{x\} \vdash_s \mathsf{new}\ x\ P \mid Q}$$

  Since $x \notin \mathrm{fn}(Q)$, we apply Lemma A.3 to $\Gamma \setminus \{\vec{x}\} \vdash_s Q$ and conclude $\Gamma \vdash_s Q$. The symmetric case is proved similarly, but using Lemma A.2 instead.

$$\dfrac{\dfrac{\Gamma \vdash_s P \quad \Gamma \vdash_s Q}{\Gamma \vdash_s P \mid Q} \qquad \Gamma \vdash_{\_} x\colon \mathsf{ch}(\_)}{\Gamma \setminus \{x\} \vdash_s \mathsf{new}\ x\ (P \mid Q)}$$

- Case $\mathsf{inaction} \equiv \mathsf{new}\ x\ \mathsf{inaction}$
  This case holds trivially by rules TP-INACT and TP-RES.

- Case $\mathsf{new}\ x\ \mathsf{new}\ y\ P \equiv \mathsf{new}\ y\ \mathsf{new}\ x\ P$
  By TP-RES rule, we derive

$$\dfrac{\dfrac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} y\colon \mathsf{ch}(\_)}{\Gamma \setminus \{y\} \vdash_s \mathsf{new}\ y\ P} \qquad \Gamma \setminus \{y\} \vdash_{\_} x\colon \mathsf{ch}(\_)}{\Gamma \setminus (\{x\} \cup \{y\}) \vdash_s \mathsf{new}\ x\ \mathsf{new}\ y\ P}$$

Thus,

$$\dfrac{\dfrac{\Gamma \vdash_s P \quad \Gamma \vdash_\_ x\colon \mathsf{ch}(\_)}{\Gamma \setminus \{x\} \vdash_s \mathsf{new}\ x\ P} \qquad \Gamma \setminus \{x\} \vdash_\_ y\colon \mathsf{ch}(\_)}{\Gamma \setminus (\{y\} \cup \{x\}) \vdash_s \mathsf{new}\ y\ \mathsf{new}\ x\ P}$$

that holds by Strengthing and Weakening.

Congruence for Networks. By induction on the typing of $\Gamma \vdash N$. We proceed by case analysis on each rule of the congruence relation defined in Figure 2, inducting on the last rule applied.

- Case $\mathsf{new}\ x\ N \mid L \equiv \mathsf{new}\ x\ (N \mid L)$, for $x \notin \mathrm{fn}(L)$.
  By hypothesis, the following type inference holds

  $$\dfrac{\dfrac{\Gamma \vdash N \quad \Gamma \vdash_\_ x \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ N} \qquad \Gamma \setminus \{x\} \vdash L}{\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ N \mid L}$$

  Since x is not in Γ, we apply clause 2 of Lemma A.3 to $\Gamma \setminus \{x\} \vdash L$, and conclude $\Gamma \vdash L$. Therefore, it is easy to show by rule TN-RES that $\Gamma \vdash \mathsf{new}\ x\ (N \mid L)$.
  The symmetric case is proved similarly, but using clause 2 of Lemma A.2, instead. The proof for $\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ (N \mid L)$ is

  $$\dfrac{\dfrac{\Gamma \vdash N \quad \Gamma \vdash L}{\Gamma \vdash N \mid L} \qquad \Gamma \vdash_\_ x \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ (N \mid L)}$$

  Since x is not in Γ, applying clause 2 of Lemma A.2 to L, we get $\Gamma \setminus \{x\} \vdash L$, and by rule TN-RES we derive

  $$\dfrac{\dfrac{\Gamma \vdash N \quad \Gamma \vdash_\_ x \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ N} \qquad \Gamma \setminus \{x\} \vdash L}{\Gamma \setminus \{x\} \vdash \mathsf{new}\ x\ N \mid L}$$

  that concludes the case.

- Case $s\{m\!\rangle \mathsf{new}\ c\ S\}[P] \equiv \mathsf{new}\ c\ s\{m\!\rangle S\}[P]\ c \notin \mathrm{fn}(P, m)$
  By hypothesis, the following type derivation holds

  $$\dfrac{\Gamma \setminus \{c\} \vdash_s m\colon \rho \quad \Gamma \setminus \{c\} \vdash_s \mathsf{new}\ c\ S \mid P \quad \Gamma \setminus \{c\} \vdash_\_ s\colon \mathsf{dom}(\rho)}{\Gamma \setminus \{c\} \vdash s\{m\!\rangle \mathsf{new}\ c\ S\}[P]}$$

  where

  $$\dfrac{\dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_\_ c\colon \mathsf{ch}(\_)}{\Gamma \setminus \{c\} \vdash_s \mathsf{new}\ c\ S} \qquad \Gamma \setminus \{c\} \vdash_s P}{\Gamma \setminus \{c\} \vdash_s \mathsf{new}\ c\ S \mid P}$$

  Since c is not in Γ, we apply the clause for Processes of Lemma A.3 to $\Gamma \setminus \{c\} \vdash_s P$, and that for Methods of the same lemma to $\Gamma \setminus \{c\} \vdash_s m$ and get, respectively, $\Gamma \vdash_s P$ and $\Gamma \vdash_s m\colon \rho$. By rules TN-DOM and TP-RES, we conclude the

case. The symmetric case is proved similarly but using the clause for Methods of Lemma A.2 and that for Processes of the same lemma. Therefore, the following derivation tree holds

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho) \qquad \dfrac{\Gamma \vdash_s S \qquad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P}}{\Gamma \vdash s\{m\!\!\rangle S\}[P]} \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\_)}{\Gamma \setminus \{c\} \vdash \mathsf{new}\ c\ s\{m\!\!\rangle S\}[P]}$$

Since $x \notin \mathrm{fn}(m)$, we apply the clause for Methods of Lemma A.2 to $\Gamma \vdash_s m\colon \rho$, and that for Processes of the same lemma to $\Gamma \vdash_s P$, and conclude the case.

- Case $s\{m\!\!\rangle S\}[\mathsf{new}\ c\ P] \equiv \mathsf{new}\ c\ s\{m\!\!\rangle S\}[P]$ for $c \notin \mathrm{fn}(m, S)$.
  By hypothesis, the following type inference holds

$$\dfrac{\Gamma \setminus \{c\} \vdash_s m\colon \rho \qquad \Gamma \setminus \{c\} \vdash_s S \mid \mathsf{new}\ c\ P \qquad \Gamma \setminus \{c\} \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \setminus \{c\} \vdash s\{m\!\!\rangle S\}[\mathsf{new}\ c\ P]}$$

  where

$$\dfrac{\Gamma \setminus \{c\} \vdash_s S \qquad \dfrac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\_)}{\Gamma \setminus \{c\} \vdash_s \mathsf{new}\ c\ P}}{\Gamma \setminus \{c\} \vdash_s S \mid \mathsf{new}\ c\ P}$$

  Since c is not in Γ, we apply the clause for Methods of Lemma A.3 to $\Gamma \setminus \{c\} \vdash_s m\colon \rho$ and that for Processes of the same lemma to $\Gamma \setminus \{c\} \vdash_s S$. By rule TP-RES it is easy to show that $\Gamma \vdash \mathsf{new}\ c\ s\{m\!\!\rangle S\}[P]$ holds.
  The symmetric case is proved as follows

$$\dfrac{\Gamma \vdash s\{m\!\!\rangle S\}[P] \qquad \Gamma \vdash_{\_} c \in \{\mathsf{ch}(\_), \mathsf{dom}(\_)\}}{\Gamma \setminus \{c\} \vdash \mathsf{new}\ c\ s\{m\!\!\rangle S\}[P]}$$

  where

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s S \qquad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\!\!\rangle S\}[P]}$$

  Since $c \notin \mathrm{fn}(m, S)$, we apply the clauses for Methods and for Processes of Lemma A.2 to $\Gamma \vdash_s m\colon \rho$ and $\Gamma \vdash_s S$, respectively, and conclude the case.

- Case $\mathsf{inaction} \equiv \mathsf{new}\ s\ s\{m\!\!\rangle \mathsf{inaction}\}[\mathsf{inaction}]$
  This case holds trivially by rules TN-INACT and TN-DOM.

◇

Lemma A.5 (Subject-Reduction)

1. *If* $\Gamma \vdash_s P$ *and* $P \to Q$, *then* $\Gamma \vdash_s Q$.
2. *If* $\Gamma \vdash N$ *and* $N \to L$, *then* $\Gamma \vdash L$.

Proof.

SR for Processes. By induction on the derivation tree for $P \to Q$. Consider the derivation tree for $\Gamma \vdash_s m\colon \rho$.

$$\frac{\Gamma \vdash_s A_1\colon \mathsf{abs}(\vec{\alpha_1}) \dots A_n\colon \mathsf{abs}(\vec{\alpha_n})}{\Gamma \vdash_s \{l_1 = A_1 \dots l_n = A_n\}\colon \{l_1\colon \vec{\alpha_1} \dots l_n\colon \vec{\alpha_n}\}}$$

- Case $c?m \mid c\,!\,l_j[\vec{V}] \to m.l_j[\vec{V}]$
  Since $\Gamma \vdash_s c?m \mid c\,!\,l_j[\vec{V}]$, then the following type derivation holds

$$\frac{\dfrac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\rho)}{\Gamma \vdash_s c?m} \qquad \dfrac{\begin{array}{c}\Gamma \vdash_s \vec{V}\colon \rho.l_j \\ \Gamma \vdash_{\_} c \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}\end{array}}{\Gamma \vdash_s c\,!\,l_j[\vec{V}]}}{\Gamma \vdash_s c?m \mid c\,!\,l_j[\vec{V}]}$$

  By definition, $m.l_j[\vec{V}]$ is $A_j\vec{V}$. $\Gamma \vdash_s A_j\vec{V}$ is

$$\frac{\Gamma \vdash_s A_j\colon \mathsf{abs}(\rho.l_j) \qquad \Gamma \vdash_s \vec{V}\colon \rho.l_j}{\Gamma \vdash_s A_j\vec{V}}$$

- Case $c?*m \mid c\,!\,l_j[\vec{V}] \to c?*m \mid m.l_j[\vec{V}]$
  The proof for $\Gamma \vdash_s c?*m \mid c\,!\,l_j[\vec{V}]$ is

$$\frac{\dfrac{\dfrac{\Gamma \vdash_s m\colon \rho}{\Gamma \vdash_{\_} c\colon \mathsf{ch}(\rho)}}{\Gamma \vdash_s c?m}}{\Gamma \vdash_s c?*m} \qquad \frac{\begin{array}{c}\Gamma \vdash_s \vec{V}\colon \rho.l_j \\ \Gamma \vdash_{\_} c \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}\end{array}}{\Gamma \vdash_s c\,!\,l_j[\vec{V}]}}{\Gamma \vdash_s c?*m \mid c\,!\,l_j[\vec{V}]}$$

  The proof for $\Gamma \vdash_s m.l_j[\vec{V}]$ holds by hypothesis.

$$\frac{\dfrac{\dfrac{\Gamma \vdash_s m\colon \rho}{\Gamma \vdash_{\_} c\colon \mathsf{ch}(\rho)}}{\Gamma \vdash_s c?m}}{\Gamma \vdash_s c?*m} \qquad \frac{\Gamma \vdash_s \vec{V}\colon \rho.l_j \qquad \Gamma \vdash_s A_j\colon \mathsf{abs}(\rho.l_j)}{\Gamma \vdash_s A_j\vec{V}}}{\Gamma \vdash_s c?*m \mid A_j\vec{V}}$$

- Case $((\vec{x})P)\vec{V} \to P[\vec{V}/\vec{x}]$
  The proof for $\Gamma \vdash_s ((\vec{x})P)\vec{V}$ is

$$\frac{\dfrac{\begin{array}{c}\Gamma \vdash_s P \\ \Gamma \vdash_{\_} s\colon \mathsf{abs}(\vec{\alpha})\end{array}}{\Gamma \setminus \{\vec{x}\} \vdash_s (\vec{x})P\colon \mathsf{abs}(\vec{\alpha})} \qquad \frac{\Gamma \setminus \{\vec{x}\} \vdash_s V_1\colon \alpha_1 \dots \Gamma \setminus \{\vec{x}\} \vdash_s V_n\colon \alpha_n}{\Gamma \setminus \{\vec{x}\} \vdash_s \vec{V}\colon \vec{\alpha}}}{\Gamma \setminus \{\vec{x}\} \vdash_s ((\vec{x})P)\vec{V}}$$

  Since $x_1\colon \alpha_1, \dots, x_n\colon \alpha_n$ and $\Gamma \vdash_s V_1\colon \alpha_1, \dots, \Gamma \vdash_s V_n\colon \alpha_n$, we meet the conditions to apply the clause for Processes of Lemma A.1 and we get $\Gamma \vdash_s P[\vec{V}/\vec{x}]$.

- Case $\dfrac{P \to Q}{P \mid R \to Q \mid R}$
  The following derivation tree holds

$$\frac{\Gamma \vdash_s P \qquad \Gamma \vdash_s R}{\Gamma \vdash_s P \mid R}$$

  Applying induction hypothesis, if $\Gamma \vdash_s P$ and $P \to Q$, then $\Gamma \vdash_s Q$. Thus, applying TP-PAR rule to $\Gamma \vdash_s Q$ and $\Gamma \vdash_s R$, we get $\Gamma \vdash_s Q \mid R$.

- Case $\dfrac{P \to Q}{\mathsf{new}\ c\ P \to \mathsf{new}\ c\ Q}$
  By hypothesis,

$$\frac{\Gamma \vdash_s P \qquad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\_)}{\Gamma \setminus c \vdash_s \mathsf{new}\ c\ P}$$

  By induction hypothesis, if $\Gamma \vdash_s P$ and $P \to Q$, then $\Gamma \vdash_s Q$, thus applying TP-RES rule we get $\Gamma \vdash_s \mathsf{new}\ c\ Q$.

- Case $\dfrac{P' \equiv P \qquad P \to Q \qquad Q \equiv Q'}{P' \to Q'}$
  By hypothesis we have $\Gamma \vdash_s P'$ and if $P' \equiv P$, then by Lemma A.4, we get $\Gamma \vdash_s P$. Since $\Gamma \vdash_s P$ and $P \to Q$, we apply induction hypothesis and get $\Gamma \vdash_s Q$. By Lemma A.4, if $\Gamma \vdash_s Q$ and $Q \equiv Q'$, then $\Gamma \vdash_s Q'$, concluding the case.

SR for Networks. By induction on the derivation tree for $N \to L$.

- Case $s\{m\rangle P\}[s\,!\,l_j[\vec{V}] \mid Q] \to s\{m\rangle(m.l_j[\vec{V}] \mid P)\}[Q]$
  The proof for $\Gamma \vdash s\{m\rangle P\}[s\,!\,l_j[\vec{V}] \mid Q]$ is

$$\frac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s P \qquad \dfrac{\Gamma \vdash_s s\,!\,l_j[\vec{V}] \mid Q}{\Gamma \vdash_s P \mid (s\,!\,l_j[\vec{V}] \mid Q)} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle P\}[s\,!\,l_j[\vec{V}] \mid Q]}$$

  where,

$$\frac{\Gamma \vdash_s \vec{V}\colon \rho.l_j \qquad \dfrac{\Gamma \vdash_{\_} s \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_s s\,!\,l_j[\vec{V}]} \qquad \Gamma \vdash_s Q}{\Gamma \vdash_s s\,!\,l_j[\vec{V}] \mid Q}$$

  Hence, the proof for $\Gamma \vdash s\{m\rangle(A_j\vec{V} \mid P)\}[Q]$ is

$$\frac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\dfrac{\Gamma \vdash_s A_j\vec{V} \mid P \qquad \Gamma \vdash_s Q}{\Gamma \vdash_s (A_j\vec{V} \mid P) \mid Q} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle(A_j\vec{V} \mid P)\}[Q]}$$

where,

$$\dfrac{\dfrac{\Gamma \vdash_s A_j : \mathsf{abs}(\rho.l_j) \quad \Gamma \vdash_s \vec{V} : \rho.l_j}{\Gamma \vdash_s A_j\vec{V}} \qquad \Gamma \vdash_s P}{\Gamma \vdash_s A_j\vec{V} \mid P}$$

- Case $s\{m\!\rangle(\mathsf{out}[r, l[\vec{V}]] \mid S)\}[P] \to r\,!\,l[s, \vec{V}] \mid s\{m\!\rangle S\}[P]$
  Since $\Gamma \vdash s\{m\!\rangle\mathsf{out}[r, l[\vec{V}]] \mid S\}[P]$, then the following type derivation holds

$$\dfrac{\Gamma \vdash_s m : \rho \qquad \dfrac{\Gamma \vdash_s \mathsf{out}[r, l[\vec{V}]] \mid S \quad \Gamma \vdash_s P}{\Gamma \vdash_s (\mathsf{out}[r, l[\vec{V}]] \mid S) \mid P} \qquad \Gamma \vdash_{\_} s : \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\!\rangle(\mathsf{out}[r, l[\vec{V}]] \mid S)\}[P]}$$

  where

$$\dfrac{\dfrac{\Gamma \vdash_{\_} r : \mathsf{dom}(\rho) \quad \Gamma \vdash_r (s\vec{V}) : \rho.l}{\Gamma \vdash_s \mathsf{out}[r, l[\vec{V}]]} \qquad \Gamma \vdash_s S}{\Gamma \vdash_s \mathsf{out}[r, l[\vec{V}]] \mid S}$$

  and the proof for $\Gamma \vdash r\,!\,l[s, \vec{V}] \mid s\{m\!\rangle S\}[P]$ is

$$\dfrac{\dfrac{\Gamma \vdash_{\_} r : \mathsf{dom}(\rho) \quad \Gamma \vdash_r (s\vec{V}) : \rho.l}{\Gamma \vdash r\,!\,l[s, \vec{V}]} \qquad \Gamma \vdash s\{m\!\rangle S\}[P]}{\Gamma \vdash r\,!\,l[s, \vec{V}] \mid s\{m\!\rangle S\}[P]}$$

  where

$$\dfrac{\Gamma \vdash_s m : \rho \qquad \dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P} \qquad \Gamma \vdash_s \_s : \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\!\rangle S\}[P]}$$

- Case $s\,!\,l_j[\vec{V}] \mid s\{m\!\rangle S\}[P] \to s\{m\!\rangle(S \mid m.l_j[\vec{V}])\}[P]$
  The proof for $\Gamma \vdash s\,!\,l_j[\vec{V}] \mid s\{m\!\rangle S\}[P]$ is

$$\dfrac{\dfrac{\Gamma \vdash_{\_} s : \mathsf{dom}(\rho) \quad \Gamma \vdash_s \vec{V} : \rho.l_j}{\Gamma \vdash s\,!\,l_j[\vec{V}]} \qquad \dfrac{\begin{array}{c}\Gamma \vdash_s m : \rho \\ \Gamma \vdash_s S \mid P \\ \Gamma \vdash_{\_} s : \mathsf{dom}(\rho)\end{array}}{\Gamma \vdash s\{m\!\rangle S\}[P]}}{\Gamma \vdash s\,!\,l_j[\vec{V}] \mid s\{m\!\rangle S\}[P]}$$

  where

$$\dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P}$$

  Since $\Gamma \vdash s\{m\!\rangle(S \mid A_j\vec{V})\}[P]$, the following proof also holds

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash_s S}{\Gamma \vdash_s A_j\vec{V}}}{\Gamma \vdash_s S \mid A_j\vec{V}} \qquad \Gamma \vdash_s P}{\Gamma \vdash_s (S \mid A_j\vec{V}) \mid P} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle(S \mid A_j\vec{V})\}[P]}$$

where,

$$\dfrac{\Gamma \vdash_s A_j\colon \mathsf{abs}(\rho.l_j) \qquad \Gamma \vdash_s \vec{V}\colon \rho.l_j}{\Gamma \vdash_s A_j\vec{V}}$$

- Case $s\{m\rangle(\mathsf{in}[P] \mid S)\}[Q] \to s\{m\rangle S\}[(Q \mid P)]$
  Since $\Gamma \vdash s\{m\rangle(\mathsf{in}[P] \mid S)\}[Q]$, the following derivation holds

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash_s P}{\Gamma \vdash_s \mathsf{in}[P]} \qquad \Gamma \vdash_s S}{\Gamma \vdash_s \mathsf{in}[P] \mid S} \qquad \Gamma \vdash_s Q}{\Gamma \vdash_s (\mathsf{in}[P] \mid S) \mid Q} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle(\mathsf{in}[P] \mid S)\}[Q]}$$

  that matches with the type derivation of $\Gamma \vdash s\{m\rangle S\}[(Q \mid P)]$

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s S \qquad \dfrac{\Gamma \vdash_s Q \qquad \Gamma \vdash_s P}{\Gamma \vdash_s Q \mid P}}{\Gamma \vdash_s S \mid (Q \mid P)} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle S\}[(Q \mid P)]}$$

- Case $s\{m\rangle(\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T)\}[Q] \to$
  $\mathsf{new}\ r\ (r\{m'\rangle S\}[P] \mid s\{m\rangle(R \mid T)\}[Q])$ if $r \notin (\mathrm{fn}(T) \cup \mathrm{fn}(Q))$
  The proof for $\Gamma \setminus r \vdash s\{m\rangle(\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T)\}[Q]$ is

$$\dfrac{\begin{array}{c}\Gamma \setminus r \vdash_s m\colon \rho \\ \Gamma \setminus r \vdash_{\_} s\colon \mathsf{dom}(\rho) \\ \Gamma \setminus r \vdash_s (\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T) \mid Q\end{array}}{\Gamma \setminus r \vdash s\{m\rangle(\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T)\}[Q]}$$

  where,

$$\dfrac{\dfrac{\Gamma \setminus r \vdash_s \mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \qquad \Gamma \setminus r \vdash_s T}{\Gamma \setminus r \vdash_s \mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T} \qquad \Gamma \setminus r \vdash_s Q}{\Gamma \setminus r \vdash_s (\mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R \mid T) \mid Q}$$

  and

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\Gamma \vdash_r S \mid P \\ \Gamma \vdash_r m'\colon \rho' \qquad \Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho')\end{array}}{\Gamma \vdash r\{m'\rangle S\}[P]} \qquad \Gamma \vdash R}{\Gamma \vdash r\{m'\rangle S\}[P] \mid R} \qquad \Gamma \vdash_{\_} r\colon \mathsf{dom}(\_)}{\Gamma \setminus r \vdash_s \mathsf{mkdom}[r, m'\rangle S, P]\ \mathsf{in}\ R}$$

from $\Gamma \vdash_r S \mid P$ we derive

$$\frac{\Gamma, r\colon \alpha \vdash_r S \qquad \Gamma, r\colon \alpha \vdash_r P}{\Gamma, r\colon \alpha \vdash_r S \mid P}$$

The proof for $\Gamma \setminus r \vdash \mathsf{new}\ r\ (r\{m'\rangle S\}[P] \mid s\{m\rangle(R \mid T)\}[Q])$ for $r \notin \mathrm{fn}(T) \cup \mathrm{fn}(Q)$ holds.

$$\frac{\dfrac{\Gamma \vdash r\{m'\rangle S\}[P] \quad \Gamma \vdash s\{m\rangle R \mid T\}[Q]}{\Gamma \vdash r\{m'\rangle S\}[P] \mid s\{m\rangle R \mid T\}[Q]} \qquad \Gamma \vdash_{\_} r \in \{\mathsf{ch}(\_), \mathsf{dom}(\_)\}}{\Gamma \setminus r \vdash \mathsf{new}\ r\ (r\{m'\rangle S\}[P] \mid s\{m\rangle R \mid T\}[Q])}$$

where,

$$\frac{\Gamma \vdash_r m'\colon \rho' \qquad \dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P} \qquad \Gamma \vdash_{\_} r\colon \mathsf{dom}(\rho')}{\Gamma \setminus r \vdash r\{m'\rangle S\}[P]}$$

and

$$\frac{\Gamma, r\colon \alpha \vdash_s m\colon \rho \qquad \dfrac{\dfrac{\Gamma \vdash_s R \quad \Gamma \vdash_s T}{\Gamma \vdash_s R \mid T} \qquad \Gamma \vdash_s Q}{\Gamma \vdash_s (R \mid T) \mid Q} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle(R \mid T)\}[Q]}$$

Since $r \notin \mathrm{fn}(T) \cup \mathrm{fn}(Q)$, we apply Lemma A.2 to $\Gamma \vdash_s m\colon \rho$, $\Gamma \vdash_s T$ and $\Gamma \vdash_s Q$, and get respectively, $\Gamma \setminus r \vdash_s m\colon \rho$, $\Gamma \setminus r \vdash_s T$, and $\Gamma \setminus r \vdash_s Q$, which concludes the case.

- Case $\dfrac{S \to T}{s\{m\rangle S\}[P] \to s\{m\rangle T\}[P]}$

  By hypothesis

  $$\frac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle S\}[P]}$$

  and applying induction hypothesis in $S \to T$ we have $\Gamma \vdash_s T$. Thus, by the TN-DOM rule we get $\Gamma \vdash s\{m\rangle T\}[P]$

- Case $\dfrac{P \to Q}{s\{m\rangle S\}[P] \to s\{m\rangle S\}[Q]}$

  By hypothesis,

  $$\frac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s S \quad \Gamma \vdash_s P}{\Gamma \vdash_s S \mid P} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle S\}[P]}$$

  Applying induction hypothesis in $P \to Q$, we have $\Gamma \vdash_s Q$. Then by TN-DOMrule we get $\Gamma \vdash s\{m\rangle S\}[Q]$.

- Case $\dfrac{N \to L}{N \mid H \to L \mid H}$
  By hypothesis,
  $$\frac{\Gamma \vdash N \qquad \Gamma \vdash H}{\Gamma \vdash N \mid H}$$
  Applying induction hypothesis in $N \mid L$, we have $\Gamma \vdash L$. Thus, by TN-PAR rule we get $\Gamma \vdash N \mid H$.

- Case $\dfrac{N \to L}{\mathsf{new}\ x\ N \to \mathsf{new}\ x\ L}$
  We derive
  $$\frac{\Gamma \vdash N \qquad \Gamma \vdash_{\_} x \in \{\mathsf{ch}(\_), \mathsf{dom}(\_)\}}{\Gamma \setminus x \vdash \mathsf{new}\ x\ N}$$
  and by induction hypothesis, if $\Gamma \vdash N$ and $N \to L$, then $\Gamma \vdash L$, thus by TN-RES rule we get $\Gamma \setminus x \vdash \mathsf{new}\ x\ L$.

- Case $\dfrac{N' \equiv N \qquad N \to L \qquad L \equiv L'}{N' \to L'}$
  By hypothesis we have $\Gamma \vdash N'$ and if $N' \equiv N$, then by Lemma A.4, we get $\Gamma \vdash N$. Since $\Gamma \vdash N$ and $N \to L$, we apply induction hypothesis and get $\Gamma \vdash L$. By Lemma A.4, if $\Gamma \vdash L$ and $L \equiv L'$, then $\Gamma \vdash_s L'$, concluding the case.

◇

B Proof of Type Safety

Theorem B.1 *If $\Gamma \vdash N$, then N is not faulty.*

Proof. We prove the contra-positive result, namely that N is faulty implies that there is no Γ such that $\Gamma \vdash N$. We separately analyse the 6 cases oin the definition of faulty.

1. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\,!\,M \mid s\{m\rangle S\}[P] \mid L)$ and $m.M$ is not defined. We assume that $\Gamma \vdash N$, and show that from this premise we may conclude that $m.M$ is defined. In fact, to conclude $\Gamma \vdash N$, we may apply rule TN-RES and derive

   $$\frac{\dfrac{\Gamma \vdash s\,!\,l[\vec{V}] \mid s\{m\rangle S\}[P] \qquad \Gamma \vdash L}{\Gamma \vdash (s\,!\,l[\vec{V}] \mid s\{m\rangle S\}[P]) \mid L} \qquad \Gamma \vdash_{\_} \vec{x} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{\vec{x}\} \vdash \mathsf{new}\ \vec{x}\ ((s\,!\,l[\vec{V}] \mid s\{m\rangle S\}[P]) \mid L)}$$

   where

   $$\frac{\dfrac{\Gamma \vdash_s \vec{V} : \rho.l \qquad \Gamma \vdash_{\_} s : \mathsf{dom}(\rho)}{\Gamma \vdash s\,!\,l[\vec{V}]} \qquad \Gamma \vdash s\{m\rangle S\}[P]}{\Gamma \vdash s\,!\,l[\vec{V}] \mid s\{m\rangle S\}[P]}$$

   and

$$\frac{\Gamma \vdash_s m\colon \rho \qquad \Gamma \vdash_s S \mid P \qquad \Gamma \vdash_\_ s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m \triangleright S\}[P]}$$

From $\Gamma \vdash_s \vec{V}\colon \rho.l$ and $\Gamma \vdash_s m\colon \rho$, we conclude that $m.M$ is defined, leading to a contradiction.

2. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\{m \triangleright P\}[s\,!\,M \mid Q] \mid L)$, and $m.M$ is not defined. By way of contradiction, we assume that $\Gamma \vdash N$. The following derivation holds by TN-RES rule

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_s m\colon \rho \quad \Gamma \vdash_\_ s\colon \mathsf{dom}(\rho) \quad \Gamma \vdash_s P \mid (s\,!\,l[\vec{V}] \mid Q)}{\Gamma \vdash s\{m \triangleright P\}[s\,!\,\vec{V} \mid Q]} \qquad \Gamma \vdash L}{\Gamma \vdash s\{m \triangleright P\}[s\,!\,l[\vec{V}] \mid Q] \mid L} \qquad \Gamma \vdash_\_ \vec{x} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{\vec{x}\} \vdash \mathsf{new}\ \vec{x}\ (s\{m \triangleright P\}[s\,!\,l[\vec{V}] \mid Q] \mid L)}$$

provided that

$$\dfrac{\Gamma \vdash_s P \qquad \dfrac{\dfrac{\Gamma \vdash_s \vec{V}\colon \rho.l \quad \Gamma \vdash_\_ s\colon \ \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_s s\,!\,l[\vec{V}]} \qquad \Gamma \vdash_s Q}{\Gamma \vdash_s s\,!\,l[\vec{V}] \mid Q}}{\Gamma \vdash_s P \mid (s\,!\,l[\vec{V}] \mid Q)}$$

where we conclude that $m.M$ is defined, leading to a contradiction.

3. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\{m \triangleright (c?m \mid c\,!\,M \mid P)\}[Q] \mid L)$, and $m.M$ is not defined. Assuming, by way of contradiction, that $\Gamma \vdash N$, we show that $m.M$ is defined by the following derivation

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash L}{\Gamma \vdash s\{m \triangleright ((c?m \mid c\,!\,l[\vec{V}]) \mid P)\}[Q]}}{\Gamma \vdash s\{m \triangleright ((c?m \mid c\,!\,l[\vec{V}]) \mid P)\}[Q] \mid L} \qquad \Gamma \vdash_\_ \{\vec{x}\} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{\vec{x}\} \vdash \mathsf{new}\ \vec{x}\ (s\{m \triangleright ((c?m \mid c\,!\,l[\vec{V}]) \mid P)\}[Q] \mid L)}$$

where

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\dfrac{\Gamma \vdash_s Q}{\Gamma \vdash_s (c?m \mid c\,!\,l[\vec{V}]) \mid P}}{\Gamma \vdash_s ((c?m \mid c\,!\,l[\vec{V}]) \mid P) \mid Q} \qquad \Gamma \vdash_\_ s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m \triangleright ((c?m \mid c\,!\,l[\vec{V}]) \mid P)\}[Q]}$$

and

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_{\_} c\colon \mathsf{ch}(\_) \quad \Gamma \vdash_s m\colon \rho}{\Gamma \vdash_s c?m} \qquad \dfrac{\Gamma \vdash_{\_} c \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\} \qquad \Gamma \vdash_s \vec{V}\colon \rho.l}{\Gamma \vdash_s c\,!\,l[\vec{V}]}}{\Gamma \vdash_s c?m \mid c\,!\,l[\vec{V}]} \qquad \Gamma \vdash_s P}{\Gamma \vdash_s (c?m \mid c\,!\,l[\vec{V}]) \mid P}$$

which concludes that $m.M$ is defined and we reach a contradiction. The proof for $c?{*}m$ is constructed along the same lines.

4. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle P\}[c?m \mid c\,!\,M \mid Q] \mid L)$, and $m.M$ is not defined.

   Assuming by way of contradiction that $\Gamma \vdash N$, we show that $m.M$ is defined by the following derivation

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash L}{\Gamma \vdash s\{m\rangle P\}[(c?m \mid c\,!\,l[\vec{V}]) \mid Q]}}{\Gamma \vdash s\{m\rangle P\}[(c?m \mid c\,!\,l[\vec{V}]) \mid Q] \mid L} \qquad \Gamma \vdash_{\_} \{\vec{x}\} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{\vec{x}\} \vdash \mathsf{new}\ \vec{x}\ (s\{m\rangle P\}[(c?m \mid c\,!\,l[\vec{V}]) \mid Q] \mid L)}$$

$$\dfrac{\Gamma \vdash_s m\colon \rho \qquad \dfrac{\Gamma \vdash_s P \qquad \dfrac{\dfrac{\Gamma \vdash_s Q}{\Gamma \vdash_s c?m \mid c\,!\,l[\vec{V}]}}{\Gamma \vdash_s (c?m \mid c\,!\,l[\vec{V}]) \mid Q}}{\Gamma \vdash_s P \mid ((c?m \mid c\,!\,l[\vec{V}]) \mid Q)} \qquad \Gamma \vdash_{\_} s\colon \mathsf{dom}(\rho)}{\Gamma \vdash s\{m\rangle P\}[(c?m \mid c\,!\,l[\vec{V}]) \mid Q]}$$

where $\Gamma \vdash_s c?m \mid c\,!\,l[\vec{V}]$ we may derive

$$\dfrac{\dfrac{\Gamma \vdash_s m\colon \rho \quad \Gamma \vdash_{\_} c\colon \mathsf{ch}(\rho)}{\Gamma \vdash_s c?m} \qquad \dfrac{\Gamma \vdash_s \vec{V}\colon \rho.l \quad \Gamma \vdash_{\_} c \in \{\mathsf{dom}(\rho), \mathsf{ch}(\rho)\}}{\Gamma \vdash_s c\,!\,l[\vec{V}]}}{\Gamma \vdash_s c?m \mid c\,!\,l[\vec{V}]}$$

from we conclude that $m.M$ is defined, leading to a contradiction. The proof for $c?{*}m$ is constructed in the same way.

5. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\rangle(((\vec{y})P)\vec{V} \mid R)\}[Q] \mid L)$, and $P[\vec{V}/\vec{y}]$ is not defined. Therefore we can deduce

$$\dfrac{\dfrac{\Gamma \setminus \{\vec{y}\} \vdash_{\_} \{\vec{x}\} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus \{\vec{y}\} \vdash s\{m\rangle(((\vec{y})P)\vec{V} \mid R)\}[Q] \mid L}}{\Gamma \setminus (\{\vec{x}\} \cup \{\vec{y}\}) \vdash \mathsf{new}\ \vec{x}\ (s\{m\rangle(((\vec{y})P)\vec{V} \mid R)\}[Q] \mid L)}$$

and

$$\dfrac{\dfrac{\dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s m \colon \rho \quad \Gamma \setminus \{\vec{y}\} \vdash_{\_} s \colon \mathsf{dom}(\rho)}{\Gamma \setminus \{\vec{y}\} \vdash_s ((\vec{y})P)\vec{V} \mid R) \mid Q} \quad \Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle((\vec{y})P)\vec{V} \mid R)\}[Q]}{\Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle((\vec{y})P)\vec{V} \mid R)\}[Q] \mid L} \quad \Gamma \setminus \{\vec{y}\} \vdash L}$$

from

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_s (\vec{y})P \colon \mathsf{abs}(\vec{\beta}) \quad \Gamma \vdash_s \vec{V} \colon \vec{\beta}}{\Gamma \vdash_s ((\vec{y})P)\vec{V}} \quad \Gamma \vdash_s R}{\Gamma \vdash_s ((\vec{y})P)\vec{V} \mid R} \quad \Gamma \vdash_s Q}{\Gamma \setminus \{\vec{y}\} \vdash_s (((\vec{y})P)\vec{V} \mid R) \mid Q}$$

where we conclude that $P[\vec{V}/\vec{x}]$ is defined, leading to a contradiction.

6. Case $N \equiv \mathsf{new}\ \vec{x}\ (s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q] \mid L)$, and $P[\vec{V}/\vec{y}]$ is not defined.

   Assuming, by way of contradiction, that $\Gamma \vdash N$, we show that $P[\vec{V}/\vec{y}]$ is defined by the following derivation

$$\dfrac{\Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q] \mid L \quad \Gamma \setminus \{\vec{y}\} \vdash_{\_} \vec{x} \in \{\mathsf{dom}(\_), \mathsf{ch}(\_)\}}{\Gamma \setminus (\{\vec{x}\} \cup \{\vec{y}\}) \vdash \mathsf{new}\ \vec{x}\ (s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q] \mid L)}$$

   and from $\Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q]$, we derive

$$\dfrac{\dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s m \colon \rho \quad \dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s Q \quad \dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s P \mid ((\vec{y})P)\vec{V}}{\Gamma \setminus \{\vec{y}\} \vdash_s P \mid (((\vec{y})P)\vec{V} \mid Q)}}{\Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q]} \quad \Gamma \setminus \{\vec{y}\} \vdash L}{\Gamma \setminus \{\vec{y}\} \vdash s\{m\!\rangle P\}[((\vec{y})P)\vec{V} \mid Q] \mid L}$$

   where the proof for $\Gamma \setminus \{\vec{y}\} \vdash_s P \mid ((\vec{y})P)\vec{V}$ is

$$\dfrac{\Gamma \setminus \{\vec{y}\} \vdash_s P \quad \dfrac{\Gamma \vdash_s \vec{V} \colon \vec{\beta} \quad \Gamma \vdash_s (\vec{y})P \colon \mathsf{abs}(\vec{\delta})}{\Gamma \setminus \{\vec{y}\} \vdash_s ((\vec{y})P)\vec{V}}}{\Gamma \setminus \{\vec{y}\} \vdash_s P \mid ((\vec{y})P)\vec{V}}$$

   Therefore, we conclude that $P[\vec{V}/\vec{y}]$ is defined, which leads to a contradiction.

$\diamond$