TyCO + Linear channels

Francisco Martins
Vasco T. Vasconcelos

DI–FCUL TR–01–11

December 2001

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

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

TyCO + Linear channels

Francisco Martins* Vasco T. Vasconcelos†

December 2001

Abstract

We present an extension to the TyCO type system that is able to identify linear channels. We prove some technical results (*e.g.* type preservation w.r.t. reduction) and present an algorithm for inferring channels usage from process expressions. Our major contribution is the inference of linear information in a calculus with recursive equations rather than replication.

1 Introduction

In this document we extend the type system for the TYped Concurrent Objects (TyCO), introduced by Vasconcelos [Vas99], to incorporate the concept of linear channel—a channel that is used just once.

In TyCO, as in π-calculus, a communication occurs when an input and an output operation is performed on the same channel. This means that a linear channel can only be used to perform at most a single input and a single output. Thus, we must count the number of inputs and outputs separately.

In addition to the monomorphic type system in [Vas99], we include *uses* that specify the number of inputs and outputs allowed on every channel. We follow [KPT99, IK00] and specify the set of uses as $\{0, 1, \omega\}$. The meaning of each element is as follows: 0 describes a channel that is never used; 1 a channel that is used at most once, and ω specifies a channel that may be used any number of times.

We also present a type system suitable for type reconstruction. This type system is parametric on a call-counting function $\mathcal{U}$ that computes channels usage in processes with recursive definitions.

*Departamento de Matemática, Universidade dos Açores.
†Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa.

One of the main motivation for this study is the inference of linear channels from process expressions. This information can be used afterwards by a compiler, at code generation phase, to produce more efficient code. TyCO is a concurrent language based on asynchronous message passing, which means that a sender does not have to wait for its message to be delivered. The implementation of this message passing style requests queues to hold undelivered messages. If one knows additional information about channels, such as linearity, we could be able to generate more efficient (both less memory and time consuming) code.

The rest of the report is organised as follows: the next section presents the syntax of TyCO process expressions. Section three introduces an extended type system, with a structural subtype relation, expressive enough to describe usage information. Section four addresses the calculus operational semantics with uses. Here we prove subject reduction. The fifth section discusses type reconstruction. We present a type system suitable for type reconstruction and an algorithm for finding principal types with uses from process expressions. In the last section we develop a function that computes, "accurately", the resources needed to typify processes with recursive definitions. In fact, the detection of linear channels in a calculus with recursive definitions constitutes the main original contribution of this report.

The proofs of claimed results are kept on a separate section for the sake of readability of the report.

2 Process expressions

Fix a denumerable set of *names*, a denumerable set of *labels*, and a denumerable set of *process variables*. We denote names, labels, and process variables, respectively, by (possibly subscribed) letters a, b, v, x, y, by letter l, and by letters X, Y. When referring to a possible empty sequence of names (or process variables) $v_1, v_2, \cdots, v_n$, we usually abbreviate it by $\tilde{v}$. The empty sequence is denoted by ε.

The syntax of process expressions is given by the following grammar.

$$
\begin{array}{rcl}
P, Q, R, S & ::= & a\,!\,l_i[\tilde{v}] \mid a\,?\,\{M\} \mid P \,|\, Q \mid \mathsf{0} \mid \mathsf{new}\ x : \rho\ P \mid X[\tilde{v}] \mid \\
 & & \mathsf{def}\ D\ \mathsf{in}\ Q \\
M & ::= & l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n \\
D & ::= & X_1\,(\tilde{x}_1) = P_1\ \mathsf{and}\ \ldots\ \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n
\end{array}
$$

Processes of the form $a\,!\,l_i[\tilde{v}]$ specify *messages*, where a is the channel through which the communication $l_i[\tilde{v}]$ is sent, l_i is a label that

selects a method in the target object and $\tilde{v}$ is the actual contents of the message. The sequence of names $\tilde{v}$ constitutes the arguments to the correspondent method.

Objects are described by processes of the form $a\,?\,\{M\}$, where a is the location of the object and M is its collection of methods. A method is of the form $l_i\,(\tilde{x}) = P_i$, where l_i is its label (unique within the collection of methods), $\tilde{x}$ is a sequence of names that represents the formal parameters, and P_i is the method body.

The process $P \mid Q$ represents the parallel execution of P and Q. *Inaction*, denoted by process $\mathsf{0}$, means a terminated process.

Scope restriction is introduced by processes of the form $\mathsf{new}\ x : \rho\ P$, suggesting x as a new channel only visible in P. We change the syntax of scope restriction (*cf.* [Vas99]), indicating explicitly the type[1] of x, in order to keep track of communications occurring privately to P. However, ρ can be inferred from P and, from a practical point of view, need not be explicitly indicated by the programmer.

Process expressions $\mathsf{def}\ X_1 = (\tilde{x}_1)P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n = (\tilde{x}_n)P_n\ \mathsf{in}\ Q$ constitute a *declaration*. *Procedures* $X_i\,(\tilde{x}_i) = P_i$ bound *process variables* X_i to processes P_i parametric on $\tilde{x}_i$. This declaration allows for mutually recursive definitions as well as several calls to X_i.

3 The extended type system

This section introduces a type system allowing for the reasoning about channel usage. For that, we must take special attention to the use of resources[2], when defining the inference rules for linear TyCO. In fact, type environments must retain information of both channels type and its usage.

3.1 Uses

In order to record the number of times that a channel has been used, we introduce the concept of *u*ses, that enables us to keep track of channels usage for input and output. To maintain a separate counting on the number of messages sent and received on a channel, we attach to each channel type a pair of *uses* (κ_1, κ_2), where κ_1 and κ_2 specify, respectively, the number of sends and receives recorded for the channel.

[1] See section 3.2 for the syntax of types.

[2] We use the terms *name* and *channel* interchangeable. However, *resource* refers also to the *capability to communicate* on a channel.

Definition 1 (Uses). *Let κ, μ range over the set of uses $\{0, 1, \omega\}$, with $0 \leq 1 \leq \omega$.*

The meaning of a use is as follows:

- 0 – it is not allowed to communicate on that channel;
- 1 – at most one communication—a linear channel;
- ω – unbound number of communications.

We define the following operations on uses that ables us to control channels usage across processes.

Definition 2 (Operations on uses). *The* sum*, the* product*, and the* least upper bound *of two uses κ_1 and κ_2, denoted respectively by $\kappa_1 + \kappa_2$, $\kappa_1 \times \kappa_2$ and $\kappa_1 \sqcup \kappa_2$, are defined as follows.*

$\kappa_1 + \kappa_2$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$\kappa_1 \times \kappa_2$	0	1	ω
0	0	0	0
1	0	1	ω
ω	0	ω	ω

$\kappa_1 \sqcup \kappa_2$	0	1	ω
0	0	1	ω
1	1	1	ω
ω	ω	ω	ω

Proposition 3. *Sum, product and least upper bound operations are commutative and associative.*

Proof. Follows directly from definition. □

Definition 4 (Suppression). *The suppression of a use κ, denoted by κ^-, is defined as*

κ^-	0	1	ω
	undef.	0	ω

3.2 Types

Types are built from a set of type variables according to the following syntax, where t denotes an arbitrary type variable and κ_1, κ_2 are uses.

$$\begin{array}{rcl} \alpha, \beta & ::= & t \quad | \quad \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\} \quad | \quad \mu t.\rho \\ \rho, \sigma, \tau & ::= & \alpha^{(\kappa_1, \kappa_2)} \end{array}$$

There are two type constructors. In the *channel type* constructor, denoted by $\{\ldots\}$, l_i represent the method labels that can be selected and $\tilde{\rho}_i$ are its types (*i.e.* the information that can flow through the channel when some l_i is selected).

The *recursive (channel) type* constructor, $\mu t.\rho$ (with $\rho \neq t$), represents the solution of the *recursive type equation* $\rho = \{\ldots, l_i : \rho, \ldots\}$, and is used to typify processes with a recursive name structure (*vide*

[VH92]). For instance, consider the process expression $a\,!\,l_1[b] \mid b\,!\,l_2[c\,a]$ and let ρ_c be the type of c. The types of the names a and b are, respectively, the solutions of the recursive type equations $\rho_a = \{l_1 : \rho_b\}^{(\kappa_a,\mu_a)}$ and $\rho_b = \{l_2 : \rho_c\,\rho_a\}^{(\kappa_b,\mu_b)}$ that can be expressed as $\mu t.\{l_1 : \{\rho_c\,t\}^{(\kappa_b,\mu_b)}\}^{(\kappa_a,\mu_a)}$ and $\mu t.\{l_2 : \rho_c\,\{l_1 : t\}^{(\kappa_a,\mu_a)}\}^{(\kappa_b,\mu_b)}$.

Uses are associated with types in the form of pairs (κ_1, κ_2), where κ_1 and κ_2 represent, respectively, the number of inputs and outputs allowed for the type.

3.3 Subtypes

We also consider a subtyping relation in our type system.

Definition 5 (Subtype). *The binary relation $\preceq$ on types is defined as the least equivalence relation closed under the following rules*

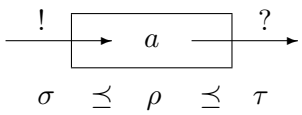
$$\frac{\kappa_1 \geq \mu_1 \geq 1 \qquad \rho_1 \preceq \tau_1 \quad \ldots \quad \rho_n \preceq \tau_n}{\{l_1 : \rho_1, \ldots, l_n : \rho_n\}^{(\kappa_1,0)} \preceq \{l_1 : \tau_1, \ldots, l_n : \tau_n\}^{(\mu_1,0)}}$$

$$\frac{\kappa_2 \geq \mu_2 \geq 1 \qquad \tau_1 \preceq \rho_1 \quad \ldots \quad \tau_n \preceq \rho_n}{\{l_1 : \rho_1, \ldots, l_n : \rho_n\}^{(0,\kappa_2)} \preceq \{l_1 : \tau_1, \ldots, l_n : \tau_n\}^{(0,\mu_2)}}$$

$$\frac{\begin{array}{cc}\kappa_1 \geq \mu_1 \geq 1 & \kappa_2 \geq \mu_2 \geq 1 \\ \tau_1 \preceq \rho_1 \quad \ldots \quad \tau_n \preceq \rho_n & \rho_1 \preceq \tau_1 \quad \ldots \quad \rho_n \preceq \tau_n\end{array}}{\{l_1 : \rho_1, \ldots, l_n : \rho_n\}^{(\kappa_1,\kappa_2)} \preceq \{l_1 : \tau_1, \ldots, l_n : \tau_n\}^{(\mu_1,\mu_2)}}$$

This means that the subtype relation $\preceq$ is covariant on the input channel arguments, contra-variant on the output channel arguments and invariant, if the channel is used both for input and output.

The idea behind channel subtyping is that we can use a channel of type ρ to send any value of *subtype* σ and the received value can be used as any *super-type* τ of ρ.



3.4 Type environments

Definition 6 (Type environment). *A type environment Γ is a mapping from names and process variables to types.*

We write $\mathrm{dom}(\Gamma)$ *for the domain of* Γ. *Type environments are ranged over by uppercase Greek letters* $\Gamma, \Delta, \Lambda, \Pi$.

The operations on uses introduced in definition 2 can be pointwise extended to types and type environments as follows

Definition 7. *Let $\star$ denote a generic operation $(+, \times, \sqcup)$. Suppose that ρ_1 and ρ_2 differ only in their outermost uses, that is $\rho_1 = \alpha^{(\kappa_1,\kappa_2)}$ and $\rho_2 = \alpha^{(\kappa_3,\kappa_4)}$. Then $\rho_1 \star \rho_2$ is defined as:*

$$\alpha^{(\kappa_1,\kappa_2)} \star \alpha^{(\kappa_3,\kappa_4)} = \alpha^{(\kappa_1 \star \kappa_3, \kappa_2 \star \kappa_4)}.$$

For type environments $\star$ is extended in the following way:

$$(\Gamma_1 \star \Gamma_2)(x) = \begin{cases} \Gamma_1(x) \star \Gamma_2(x), & \text{if } x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2) \\ \Gamma_1(x), & \text{if } x \in \mathrm{dom}(\Gamma_1) \setminus \mathrm{dom}(\Gamma_2) \\ \Gamma_2(x), & \text{if } x \in \mathrm{dom}(\Gamma_2) \setminus \mathrm{dom}(\Gamma_1) \end{cases}$$

We now introduce some operations on type environments that are needed for the rest of the report.

Definition 8 (Operations on type environment). *Let Γ be a type environment, a a name, $\rho = \alpha^{(\kappa_1,\kappa_2)}$ be a type, and μ a use constant.*

1. *$\Gamma + a : \rho$ denotes the type environment Δ where $\mathrm{dom}(\Delta) = \mathrm{dom}(\Gamma) \cup \{a\}$, $\Delta(x) = \Gamma(x)$ for all $x \neq a$, and*

$$\Delta(a) = \begin{cases} \Gamma(a), & \text{if } a \notin \mathrm{dom}(\Gamma), \\ \Gamma(a) + \rho, & \text{if } a \in \mathrm{dom}(\Gamma) \text{ and } + \text{ is defined}, \\ \text{undefined} & \text{if } a \in \mathrm{dom}(\Gamma) \text{ and } + \text{ is undefined}. \end{cases}$$

2. *When $a \notin \mathrm{dom}(\Gamma)$, we define $\Gamma \uplus a : \rho$ as the type environment Δ where $\mathrm{dom}(\Delta) = \mathrm{dom}(\Gamma) \cup \{a\}$, $\Delta(x) = \Gamma(x)$ for all $x \in \mathrm{dom}(\Gamma)$, and $\Delta(a) = \rho$.*
3. *$\Gamma \backslash a$ denotes the type environment whose domain is $\mathrm{dom}(\Gamma) \backslash \{a\}$.*
4. *$\mu \times \Gamma$ denotes the type environment Δ where $\mathrm{dom}(\Delta) = \mathrm{dom}(\Gamma)$ and $\Delta(x) = \alpha^{(\mu \times \kappa_1, \mu \times \kappa_2)}$ for all x such that $\Gamma(x) = \alpha^{(\kappa_1,\kappa_2)}$.*

3.5 Counting procedure calls

The def construct binds processes to process variables and allows for calls to these processes within its scope. The number of calls to a procedure is unrestricted, meaning that a procedure can be called any number of times.

For a process P to be typified correctly, the input and output uses of every (type of every) name on P must reflect, at least, its communication capabilities. If a name, say a, occurs free in a procedure $X\,(\tilde{v}) = P$, it is not enough to consider only the usage of a within P. In fact, the usage of a depends also from the number of times that X

is called within a given process. To illustrate this situation consider the following example.

$$\mathsf{def}\ X\,(v) = a\,!\,l_1[v]\ \mathsf{in}\ X[b]\,|\,X[c].$$

Considering $a\,!\,l_i[\tilde{v}]$, we may say that the usage of a is $(0,1)$. But X is called twice in the scope of the definition. Then the usage of channel a must be at least $(0,\omega)$.

We propose a function $\mathcal{U}(X,D,Q)$ that computes the number of times that X is called in Q, regarding the collection of definitions D. The call to $\mathcal{U}(X, X\,(v) = a\,!\,l_1[v], X[b]\,|\,X[c])$, must yield ω, since X is used twice in Q, and thus the usage of a is $\omega \times (0,1) = (0,\omega)$.

As a name may have more then one type, it is plausible to expect that there is not a unique way to compute the number of calls to X. Trivially the constant function $W(X,D,Q) = \omega$ will do the job, since $\alpha^{(\omega,\omega)} \preceq \alpha^{(\kappa_1,\kappa_2)}$, for any use κ_1, κ_2. But then the question is: is the function interesting enough from a practical point of view? Certainly not, because this information cannot help us in any kind of compiler optimisation. We have devised a function, introduced in section 6, that more accurately computes the number of times $(0,1,\omega)$ that a process variable is called in a given process.

In what follows we state the properties that a function must satisfy in order to be a *call-counting function*.

Definition 9 (Call counting function). *Let $D \stackrel{\text{def}}{=} X_1\,(\tilde{x}_1) = P_1$ and $\ldots$ and $X_n\,(\tilde{x}_n) = P_n$. A function $\mathcal{U}$ is a* call-counting function *if it satisfies the following requirements.*

1. $\mathcal{U}(X,D,Q) \geq \mathcal{U}(X,D,R)$, *if* $Q \stackrel{\ell}{\rightarrow} R$, *for all* ℓ,
2. $\mathcal{U}(X,D,X_i[\tilde{v}]\,|\,Q) = 1 + \mathcal{U}(X,D,\{\tilde{v}/\tilde{x}_i\}P_i\,|\,Q)$, *if* $X = X_i$ *for some* i,
3. $\mathcal{U}(X,D,X_i[\tilde{v}]\,|\,Q) = \mathcal{U}(X,D,\{\tilde{v}/\tilde{x}_i\}P_i\,|\,Q)$, *if* $X \neq X_i$ *for no* i.

The above assertions define the behaviour of $\mathcal{U}$ during process reduction[3]. The first assertion states that the number of potential calls to a particular procedure cannot increase during reduction. Assertions (2) and (3) refer specifically to reductions that occur on a call: if it is on X—the variable that we are counting—then the number of calls decreases by 1, because X is called in P_i the same number of times in each equation side, plus one more time in the call to $X[\tilde{v}]$ itself. Otherwise (assertion 3) the number of potential calls to X is not affected.

[3]See section 4.2 for the reduction relation.

Proposition 10. *The constant function W, defined as $W(X, D, Q) = \omega$, is a call counting function.*

Proof. Trivial from $\omega \geq \omega$ and $\omega = 1 + \omega$. □

3.6 Typing rules

We now are in position to present the typing rules for linear TyCO.

Notice that we pay special attention to type environments used to typify processes, and construct them in such a way that they contain both typing information on what channels transport (the number and types of the names that the channel transports) and channels usage (the number of times that the channel may be used for input and for output).

A judgement is an expression of the form $\Gamma \vdash P$ and means, not only that P is correctly typed under Γ, but also that the resources are used according to the uses specified by the types.

$$\textsc{Msg} \quad a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i \vdash a\,!\,l_i[\tilde{v}]$$

Rule Msg expresses the fact that a must be a channel with, at least, output capabilities. We add $\tilde{v} : \tilde{\rho}_i$ to the type environment to take into account the usage of $\tilde{v}$ by the receiver.

$$\textsc{Obj} \quad \frac{\Gamma \vdash M : \alpha}{\Gamma + a : \alpha^{(1,0)} \vdash a\,?\,\{M\}}$$

Following the same approach, a must at least have input capabilities.

$$\textsc{Res} \quad \frac{\Gamma \uplus x : \rho \vdash P}{\Gamma \vdash \mathsf{new}\ x : \rho\ P}$$

The restriction rule specifies that the corresponding binding variable moves from the type environment to the new operator.

$$\textsc{Par} \quad \frac{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash Q}{\Gamma_1 + \Gamma_2 \vdash P \,|\, Q}$$

The Par rule establishes that the usage of channels in the process $P \,|\, Q$ is the *sum* of the two environments Γ_1 and Γ_2 that typify P and Q, respectively. This means that the resources consumed by $P \,|\, Q$ are those consumed by P together with those consumed by Q. As an example, suppose that $P \stackrel{\text{def}}{=} a\,?\,\{l_1\,(x) = a\,!\,l_1[x]\}$ and $Q \stackrel{\text{def}}{=} a\,!\,l_1[v]$. Then a is used both for input and for output (once) in P and used again in Q, that is, a is no longer a linear channel in $P \,|\, Q$.

$$\textsc{App}\quad X : \rho_1 \ldots \rho_n \uplus v_1 : \rho_1 + \ldots + v_n : \rho_n \vdash X\,[\tilde{v}]$$

The name sequence $\tilde{v}$ constitutes the arguments of X. Therefore, the sum $v_1 : \rho_1 + \ldots + v_n : \rho_n$ enables us to keep track of the usage of v_i in the process bound to X. Notice that v_i are not necessary distinct.

$$\textsc{Nil}\quad \emptyset \vdash \mathbf{0}$$

The inaction process needs no resources to be typified.

$$\textsc{Def}\quad \frac{\Gamma_1 \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \quad \cdots \quad \Gamma_n \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n \qquad \Delta \uplus X_i : \tilde{\rho}_i \vdash Q}{\sum_i \mathcal{U}(X_i, D, Q) \times \Gamma_i + \Delta \vdash \mathsf{def}\ X_1 = (\tilde{x}_1)\ P_1 \ldots\ \mathsf{in}\ Q}$$

The product of each Γ_i by $\mathcal{U}(X_i, D, Q)$—the number of times that X_i is called in $\mathsf{def}\ D\ \mathsf{in}\ Q$—accounts for the fact that some of the X_i can be called zero or more times; each time X_i is called we need the resources specified by Γ_i to be available.

$$\textsc{Meth}\quad \frac{\Gamma \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \qquad \ldots \qquad \Gamma \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n}{\Gamma \vdash l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}}$$

In order to typify a collection of methods we do not add or multiply the resulting environment, since just one of the methods is active at a given time. Thus, we use as the result environment the most consuming resources of each method (because Γ is shared).

$$\textsc{Sub}\quad \frac{\Gamma \uplus x : \rho' \vdash P \qquad \rho \preceq \rho'}{\Gamma \uplus x : \rho \vdash P}$$

$$\textsc{Weak}_1\quad \frac{\Gamma \vdash P}{\Gamma + x : \rho \vdash P} \qquad \textsc{Weak}_2\quad \frac{\Gamma \vdash P}{\Gamma \uplus X : \tilde{\rho} \vdash P}$$

These rules Sub, $\textsc{Weak}_1$ and $\textsc{Weak}_2$ are the usual rules for subsumption and weakening.

3.7 Examples

We illustrate the typing rules with some examples.

Example 11. The process $a\,!\,l_1[]$ is typified with the type environment $\{a : \{l_1 : \varepsilon\}^{(0,1)}\}$, where ε denote the empty type sequence. This means that name a has a channel type with the capability of performing one output and no input at all. Types are not uniquely determined, for instance, other type possibilities for name a include $\{l_1 : \varepsilon\}^{(1,1)}$ and $\{l_1 : \varepsilon\}^{(\omega,1)}$.

Example 12. In the process $a\,?\,\{l_1\,() = x\,!\,l[], l_2\,(y) = x\,!\,l[]\}$, name a has type $\{l_1 : \varepsilon, l_2 : \{\}^{(0,0)}\}^{(1,0)}$, meaning that it has only input capabilities, and that y has no communication capability at all. On the other hand, x has type $\{l : \varepsilon\}^{(0,1)}$, despite the fact that it appears twice in output position. This is so because x is used in distinct methods and we compute its least upper bound usage. Other type possibilities for name a include $\{l_1 : \varepsilon, l_2 : \{\}^{(0,0)}\}^{(\omega,\omega)}$ and $\{l_1 : \varepsilon, l_2 : \{\}^{(1,\omega)}\}^{(1,1)}$.

Example 13. Consider the following process expression

$$a\,?\,\{l_1\,(x) = x\,!\,l[]\}\,|\,a\,!\,l_1[y]\,|\,y\,?\,\{l = ()\,\mathsf{0}\}.$$

Name x has type $\{l : \varepsilon\}^{(0,1)}$, because it is used for output and the method with label l carries nothing. Hence, name a has type $\{l_1 : \{l : \varepsilon\}^{(1,0)}\}^{(1,1)}$, because it is both used at an input and an output position. The type of y can be a little bit tricky, since it is used for output (implicitly) on $a\,!\,l_i[y]$ and used for input (explicitly) on $y\,?\,\{l\,() = \mathsf{0}\}$. So, y has type $\{l : \varepsilon\}^{(1,1)}$.

Example 14. Consider the process $\mathsf{def}\ D\ \mathsf{in}\ Q$ where

$$\begin{aligned} D &\stackrel{\text{def}}{=} X_1\,(x) = a\,!\,l_1[x]\ \mathsf{and}\ X_2\,(y) = X_3[]\,|\,X_1[y]\ \mathsf{and} \\ &\qquad X_3\,() = X_4[]\ \mathsf{and}\ X_4\,() = X_3[]\ \mathsf{and}\ X_5\,(x) = a\,!\,l_1[x] \\ Q &\stackrel{\text{def}}{=} X_2[b]\,|\,a\,?\,\{l_1\,(z) = z\,!\,l_2[]\} \end{aligned}$$

Name a has type $\{l_1 : \{l_2 : \varepsilon\}^{(0,1)}\}^{(1,1)}$, since it is used for input in Q, for output in the call to X_2, and transports name b. Name b has type $\{l_2 : \varepsilon\}^{(0,1)}$ because it is used as an argument to the method labelled l_1 locate at a. Notice however that, despite the fact that X_3 and X_4 are defined recursively, X_1 is only called once from X_2 (X_1 is not reachable from X_3 or X_4). Nevertheless, $W(X_i, D, Q) = \omega$, for $1 \leq i \leq 5$.

Example 15. In the following def process a and b are free names and X_2 is defined recursively

$$\mathsf{def}\ X_1\,(x) = a\,!\,l_1[x]\ \mathsf{and}\ X_2\,() = X_2[]\,|\,X_1[y]\ \mathsf{in}\ X_2[].$$

Notice that x has an undetermined usage, since it is not possible to compute its use within the object located at a. Thus, y has also an undetermined usage. For the sake of this example we set their uses pair to $(0, 0)$. Later, we explain how to deal with undetermined usages (through *use variables*). Name a has type $\{l_1 : t^{(0,0)}\}^{(0,\omega)}$ because X_2 is reachable from the body of the def (and is recursive) and X_1 is also reachable from X_2. Hence, we have to compute $\omega \times \{l_1 : t^{(0,0)}\}^{(0,1)}$.

4 Reduction semantics (with uses)

The operational semantics of the calculus is presented following Milner [MPW92]. We first define a *congruence relation* between processes that simplifies the *reduction relation* introduced thereafter.

Free and bound names of P, denoted by $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$, respectively, are defined in the usual way; that is, bound names are introduced by two process constructions: (1) procedures—$X\,(\tilde{x}) = P$—the sequence of names $\tilde{x}$ are bound in P; (2) restrictions—$\mathsf{new}\ x : \rho\ P$—name x is also bound in P. Every name occurring in a process that is not bound is free. For process variables we say that X is free, if it is not in the scope of a def process. Otherwise X is bound. The sets of free and bound process variables are denoted, respectively, by $\mathrm{fv}(P)$ and $\mathrm{bv}(P)$.

Notice that on a declaration, say $X(\tilde{v}) = P$, the free names of P are not necessary in $\tilde{v}$. Name a in $X(v) = a\,!\,l_i[v]$ illustrates this situation.

Both for names and process variables, we follow the variable convention (as in lambda calculus), and α-convert the bound names (variables) of a process in such a way that every bound name (variable) is different from the other free and bound names (variables) of the process. The exact definition of α-conversion, $\equiv_\alpha$, is in the appendix (definition 45).

4.1 Structural congruence

The structural congruence relation is the least congruence on process expressions closed under the following rules:

1. $P \equiv Q$, if P is α-convertible to Q,
2. $P \,|\, Q \equiv Q \,|\, P, \quad (P \,|\, Q) \,|\, R \equiv P \,|\, (Q \,|\, R), \quad P \,|\, \mathsf{0} \equiv P,$
3. $\mathsf{new}\ x : \rho\ \mathsf{0} \equiv \mathsf{0}$,
   $\mathsf{new}\ x : \rho\ \mathsf{new}\ y : \sigma\ P \equiv \mathsf{new}\ y : \sigma\ \mathsf{new}\ x : \rho\ P$, if $x \neq y$ or (when $x = y$) $\sigma \preceq \rho$,
   $(\mathsf{new}\ x : \rho\ P) \,|\, Q \equiv \mathsf{new}\ x : \rho\ (P \,|\, Q)$, if $x \notin \mathrm{fn}(Q)$,
4. $\mathsf{def}\ D\ \mathsf{in}\ \mathsf{0} \equiv \mathsf{0}$,
   $\mathsf{def}\ D\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q \equiv \mathsf{new}\ x : \rho\ \mathsf{def}\ D\ \mathsf{in}\ Q$, if $x \notin \mathrm{fn}(D)$,
   $(\mathsf{def}\ D\ \mathsf{in}\ Q) \,|\, R \equiv \mathsf{def}\ D\ \mathsf{in}\ (Q \,|\, R)$, if $\{X_i\} \cap \mathrm{fv}(R) = \emptyset$ and $D \stackrel{\mathrm{def}}{=} X_1\,(\tilde{x}_1) = P_1\ \mathsf{and}\ \ldots\ \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n$.
5. $M_1 \equiv M_2$, if M_2 is a reordering of methods defined in M_1.
   $D_1 \equiv D_2$, if D_2 is a reordering of processes defined in D_1.

Notice, in the second clause of the fourth rule, that the scope of a new can be extended to embrace a def process, provided that x is not free in any abstraction of D. We must set this condition since, for each procedure $X_i\,(\tilde{v}_i) = P_i$, the free names of P_i may not all belong to the name sequences $\tilde{v}_i$.

Also notice, in third clause of the fourth rule, that a process can be included or not in the scope of a def, if no X_i is in $\mathrm{fv}(R)$.

Another comment, perhaps more subtle, is the side condition on new rule when restricted names commute (clause 2, rule 3). In fact, if the restricted name is the same for both new processes, then the occurrences of the restricted name in P are bound to the inner new, and when we commute the name restrictions we have to guarantee that the type of the "new" restricted name offers, at least, the same capabilities as the previous one, *i.e.*, it must be its subtype (for details see the proof of lemma 16 in appendix A).

The intuitive meaning of structural congruence on processes is that whenever P is part of a process and $P \equiv Q$, then we can replace P by Q without affecting the behaviour of the process. Thus, it seems natural that the typing environment of P must be the same as Q. The following result makes this statement precise.

Lemma 16. *If* $\Gamma \vdash P$ *and* $P \equiv Q$*, then* $\Gamma \vdash Q$*.*

Proof. See appendix A. □

4.2 Reduction relation

We now present the one step reduction relation with uses. Each reduction is labelled either with a free channel x, or with the special symbol ϵ denoting a communication on a bound channel or a process instantiation. We use ℓ to range both over free names and ϵ. The expression $\{\tilde{v}/\tilde{x}\}P$ denotes the simultaneous substitution of free occurrences (in P) of pairwise distinct names x_i for v_i.

$$\textsc{Com}\quad a\,!\,l_i[\tilde{v}]\,|\,a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\} \xrightarrow{a} \{\tilde{v}/\tilde{x}_i\}P_i$$

This is the basic *communication* rule between a *message* and an *object*. The resulting process is the method body P_i, selected by the label l_i, with its parameters $\tilde{x}_i$ replaced by the arguments $\tilde{v}$.

$$\textsc{Par}\quad \frac{P \xrightarrow{\ell} R}{P\,|\,Q \xrightarrow{\ell} R\,|\,Q}$$

The PAR rule allows a reduction to occur inside a parallel composition of processes.

$$\textsc{Res}_1 \ \frac{P \xrightarrow{x} R}{\mathsf{new}\ x : \alpha^{(\kappa_1,\kappa_2)} P \xrightarrow{\epsilon} \mathsf{new}\ x : \alpha^{(\kappa_1^-,\kappa_2^-)} R}$$

$$\textsc{Res}_2 \ \frac{P \xrightarrow{\ell} R \qquad \ell \neq x}{\mathsf{new}\ x : \rho\ P \xrightarrow{\ell} \mathsf{new}\ x : \rho\ R}$$

We distinguish two cases for reduction inside a restriction: (1) $\textsc{Res}_1$ represents a reduction that occurs on the restricted name[4]. In this case we label the reduction with ϵ (communication over a bound channel) and indicate explicitly, in the type of x, that it was used in the reduction; (2) $\textsc{Res}_2$ denotes a reduction occurs on a name other than x.

$$\textsc{Def} \ \frac{P \xrightarrow{\ell} Q}{\mathsf{def}\ D\ \mathsf{in}\ P \xrightarrow{\ell} \mathsf{def}\ D\ \mathsf{in}\ Q}$$

$$\begin{aligned}\textsc{Call}\ \ & \mathsf{def}\ D\ \mathsf{and}\ X\,(\tilde{x}) = P\ \mathsf{in}\ X[\tilde{v}] \,|\, Q \xrightarrow{\epsilon} \\ & \mathsf{def}\ D\ \mathsf{and}\ X\,(\tilde{x}) = P\ \mathsf{in}\ \{\tilde{v}/\tilde{x}\}P \,|\, Q, \\ & \text{where } D \stackrel{\text{def}}{=} X_1\,(\tilde{x}_1) = P_1\ \mathsf{and}\ \ldots\ \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\end{aligned}$$

The two rules above define the behaviour of a def process. The DEF rule specifies that a communication can occur inside a def process. CALL rule describes the replacement of a process variable by its definition, performing the necessary substitution. Notice, however, that this *procedure call* does not consume any channel resources, hence the transition is labelled by ϵ.

$$\textsc{Str} \ \frac{P \equiv R \qquad R \xrightarrow{\ell} S \qquad S \equiv Q}{P \xrightarrow{\ell} Q}$$

This last rule incorporates structural congruence into reduction.

Intuitively the *reduction relation with uses* just introduced, allows the control of communications over channels in the following way: if a process P is typified by some environment, say Γ, and P can reduce by ℓ to become Q, then the environment resulting from Γ by suppressing the capabilities consumed in the reduction typify Q. More precisely, if $\Gamma \vdash P$ and $P \xrightarrow{\ell} Q$, then $\Gamma^{-\ell} \vdash Q$. This result (subject reduction—theorem 19) constitutes the goal of the current section.

The effect of consuming resources in a reduction is made precise by the following definition:

[4]Recall the definition of suppression (definition 4, page 4).

Definition 17. *The type environment* $\Gamma^{-\ell}$ *is obtained from* Γ *as follows.*

$$\Gamma^{-\ell}(a) = \begin{cases} \Gamma(a), & \text{if } a \neq \ell, \\ \alpha^{(\kappa_1^-, \kappa_2^-)}, & \text{if } \Gamma(a) = \alpha^{(\kappa_1, \kappa_2)} \text{ and } \kappa_1^-, \kappa_2^- \text{ defined}, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Notice that $a \neq \ell$ implies that $a \neq \epsilon$, since ϵ never belongs to $\mathrm{dom}(\Gamma)$.

Proposition 18. *If* $\Gamma^{-\ell} \vdash P$*, then* $\Gamma \vdash P$*.*

Proof. The case when $\ell \notin \mathrm{dom}(\Gamma)$ is trivial, since $\Gamma^{-\ell} = \Gamma$. On the other hand, if $\ell \in \mathrm{dom}(\Gamma)$, we know that $\Gamma^{-\ell} \vdash P$ and that $\Gamma(\ell) \preceq \Gamma^{-\ell}(\ell)$. Applying Sub rule

$$\textsc{Sub}\ \frac{\Gamma^{-a} \vdash P \qquad \Gamma(\ell) \preceq \Gamma^{-\ell}(\ell)}{\Gamma \vdash P}$$

concludes the proof. □

The following result asserts that the types of free names remain invariant during reduction. In the present case, where types have uses pairs associated, the result goes further and specifies the variation on the uses of the name where the reduction took place.

Theorem 19 (Subject reduction). *If* $\Gamma \vdash P$ *and* $P \xrightarrow{\ell} Q$*, then* $\Gamma^{-\ell}$ *is defined and* $\Gamma^{-\ell} \vdash Q$*.*

Proof. See appendix A. □

The usual subject reduction result can be obtained as a consequence of the previous theorem.

Corollary 20. *If* $\Gamma \vdash P$ *and* $P \xrightarrow{\ell} Q$*, then* $\Gamma \vdash Q$*.*

Proof. From hypothesis and theorem 19, we find that $\Gamma^{-\ell} \vdash Q$. Applying proposition 18 one concludes the proof. □

5 Type reconstruction

This section addresses type reconstruction. We want to recover types from processes without any explicit annotations from the programmer, and we want to identify (on a systematic basis) the linear channels in a program. This kind of information, achieved by a static analysis on processes, is quite important at compile time, namely on queue allocation. If a channel is linear, then we need only to allocate a memory

cell to manage the communications within this channel, because one has the guarantee that there will not ever be more than one message waiting to be received for that channel. Furthermore, this memory can be deallocated after the communication takes place.

Our approach follows Igarashi and Kobayashi [IK00]: first we devise a new typing system, equivalent to the previous one (see section 3.6), but suited for type reconstruction; then we present some operators to compute type restrictions, and proceed with the presentation of an algorithm that computes a principal typing and a set of restrictions to uses. For the resolution of this special kind of restrictions see [IK00].

5.1 Type constraints

We extend uses syntax to incorporate use variables. Type and use variables play and important role in type reconstruction.

Definition 21 (Use expressions). *Let u (possible subscribed) range over an infinite set of use variables. The set of use expressions is given by the following syntax.*

$$\kappa ::= 0 \mid 1 \mid \omega \mid u \mid \kappa_1 + \kappa_2 \mid \kappa_1 \cdot \kappa_2 \mid \kappa_1 \sqcup \kappa_2.$$

0, 1, ω are called use constants.

We keep information about type and use variables as subtyping constraints on a set C.

Definition 22 (Subtype restriction set). *A* subtype restriction set *C is a set of subtype expressions $\rho_1 \preceq \rho_2$, called* restriction expressions. *We extend $\preceq$ to type environments, and let $\Gamma \preceq \Delta$ denote the subtype restriction set $\{\Gamma(x) \preceq \Delta(x) \mid x \in \mathrm{dom}(\Delta)\}$.*

The notion of type/use substitution is simpler than variable substitution, since there are no bound variables in type/use expressions.

Definition 23 (Substitution, instance). *A substitution S is an expression*

$$\{\rho_1/t_1, \ldots, \rho_n/t_n, \kappa_1/u_1, \ldots, \kappa_m/u_m\},$$

where $t_1, \ldots, t_n$ are distinct type variables, $\rho_1, \ldots, \rho_n$ are types ($n \geq 0$), $u_1, \ldots, u_m$ distinct use variables, and $\kappa_1, \ldots, \kappa_m$ uses ($m \geq 0$). The domain of S, denoted $\mathrm{dom}(S)$*, is the set $\{t_1, \ldots, t_n, u_1, \ldots, u_m\}$. For any type ρ define $S\rho$ to be the type obtained by simultaneously substituting ρ_1 for $t_1, \ldots, \rho_n$ for t_n and κ_1 for $u_1, \ldots, \kappa_m$ for u_m throughout ρ. We call $S\rho$ an* instance *of ρ.*

We extend substitution S to finite sequences of types $\tilde{\rho}$, to type environments Γ, to subtype restriction sets C, and also to processes P. Moreover, if SC contains no type/use variables, we call S a ground substitution *on C.*

The ground substitutions that interest us are those that turn true all the subtype restrictions of a subtype constraint set.

Definition 24 (Solution of a constraint set). *The substitution S is a* ground solution *of C, if $S\rho_1 \preceq S\rho_2$ holds for every restriction expression $\rho_1 \preceq \rho_2$ in C.*

The next definition relates subtype constraint sets.

Definition 25. *The constraint set C_1 satisfies C_2, denoted by $C_1 \models C_2$, if every solution of C_1 is also a solution of C_2.*

Sometimes we use the expression "C_1 is a subtype constraint stronger than C_2".

Proposition 26. *$C \models C_1$ and $C \models C_2$, if, and only if, $C \models C_1 \cup C_2$.*

Proof. Let S be a solution of C. Since $C \models C_1$ and $C \models C_2$, S satisfies every restriction of C_1 and C_2, that is, S satisfies $C_1 \cup C_2$. Conversely, if S satisfies $C_1 \cup C_2$, then S satisfies C_1 and satisfies C_2. □

5.2 Type system for reconstruction

We introduce a syntax-directed typing system that identify linear channels. These typing rules are adequate to perform type inference because at each step in the inference there is exactly one rule to apply, whereas in the type system devised at section 3.6 we could apply Sub and/or Weak at any inference step. (Nevertheless, the former type system is best suited for proving the subject reduction theorem.)

Judgements are of the form $\Gamma; C \vdash P$ with the intended meaning that $S\Gamma \vdash SP$ holds for any substitution S that is a solution of C.

$$\textsc{Msg}_{\text{SD}} \quad \frac{C \models \Gamma \preceq a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i}{\Gamma; C \vdash a\,!\,l_i[\tilde{v}]}$$

$$\textsc{Obj}_{\text{SD}} \quad \frac{\Delta; C' \vdash M : \alpha \qquad C \models C' \qquad C \models \Gamma \preceq \Delta + a : \alpha^{(1,0)}}{\Gamma; C \vdash a\,?\,\{M\}}$$

$$\textsc{Res}_{\text{SD}} \quad \frac{\Gamma \uplus x : \sigma; C \vdash P \qquad C \models \rho \preceq \sigma}{\Gamma; C \vdash \mathsf{new}\ x : \rho\ P}$$

$$\textsc{Par}_{\text{SD}}\ \frac{\Gamma_1; C_1 \vdash P \quad \Gamma_2; C_2 \vdash Q \quad C \models C_1 \cup C_2 \cup \Gamma \preceq \Gamma_1 + \Gamma_2}{\Gamma; C \vdash P \mid Q}$$

$$\textsc{App}_{\text{SD}}\ \frac{C \models \tilde{\sigma} \preceq \tilde{\rho}}{\Gamma \uplus X : \tilde{\rho} \uplus \tilde{v} : \tilde{\sigma}; C \vdash X[\tilde{v}]} \qquad \textsc{Nil}_{\text{SD}}\ \ \Gamma; \emptyset \vdash \mathsf{0}$$

$$\textsc{Def}_{\text{SD}}\ \frac{\begin{array}{c} \biguplus_j X_j : \tilde{\rho}_j \uplus \Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i; C_i \vdash P_i, \quad 1 \leq i \leq n \\ \biguplus_j X_j : \tilde{\rho}_j \uplus \Delta; C' \vdash Q \\ C \models \Gamma \preceq \sum_j \mathcal{U}(X_i, D, Q) \times \Gamma_j + \Delta \\ C \models \bigcup_j (C_j \cup \{\tilde{\sigma}_j \preceq \tilde{\rho}_j\}) \cup C' \end{array}}{\Gamma; C \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\ \mathsf{in}\ Q}$$

$$\textsc{Meth}_{\text{SD}}\ \frac{\begin{array}{c} \Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i; C_i \vdash P_i, \quad 1 \leq i \leq n \\ C \models \bigcup_j (\{\Gamma \preceq \Gamma_j\} \cup C_j \cup \{\tilde{\rho}_j \preceq \tilde{\sigma}_j\}) \end{array}}{\Gamma; C \vdash l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}}$$

Notice that there are no rules for Sub and Weak, since every rule combines the effect of these two. If fact, the preceding rules reflect the effect of the rules Weak and Sub applied together, and are obtained from the rules in section 3.6 with the same name, but without subscript.

The equivalence between the type system just presented and the one of subsection 3.6 is made precise by the following theorem.

Theorem 27 (Equivalence of the two typing systems).

1. *Suppose that* $\Gamma; C \vdash P$*. If* S *is a solution of* C *and its domain includes all type/use variables in* Γ *and in* P*, then* $S\Gamma \vdash SP$*;*
2. *If* $\Gamma \vdash P$*, then* $(\Gamma, \emptyset) \vdash P$*.*

Proof. See appendix A. □

Proposition 28. *If* $\Gamma; C \vdash P$ *and* $\Delta; C'$ *is an instance of* $\Gamma; C$*, then* $\Delta; C' \vdash P$*.*

As was mentioned before (see section 3) types are not uniquely determined. The next definition relates such types.

Definition 29 (Principal typing). *The pair* $\Gamma; C$ *is a principal typing of* P*, if*

1. $\Gamma; C \vdash P$*, and*
2. *If* $\Delta; C' \vdash P$*, then* $\Delta; C'$ *is an instance of* $\Gamma; C$*.*

5.3 Type reconstruction algorithm

The reconstruction of principal typings proceeds in two steps: (1) compute a set of constraints for both type and use variables; (2) resolve these constraints. We address step one. For constraint resolution see reference [IK00].

First, we define the operators $\oplus, \sqcup$, and $\otimes$, intended to compute the most general pair typing.

Definition 30. $\Gamma_1 \oplus \Gamma_2$ *is a pair* $\Gamma; C$ *defined as follows.*

$$\begin{aligned}\Gamma &= \{x : \beta^{(r_x,u_x)} \,|\, x \in \mathrm{dom}(\Gamma_1) \cup \mathrm{dom}(\Gamma_2)\},\\ C &= C_1 \cup C_2 \cup \{\Gamma(x) \preceq \Delta_1(x) + \Delta_2(x) \,|\, x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)\}\\ &\qquad \cup \{\Gamma(x) \preceq \Gamma_1(x) \,|\, x \in \mathrm{dom}(\Gamma_1) \setminus \mathrm{dom}(\Gamma_2)\}\\ &\qquad \cup \{\Gamma(x) \preceq \Gamma_2(x) \,|\, x \in \mathrm{dom}(\Gamma_2) \setminus \mathrm{dom}(\Gamma_1)\},\end{aligned}$$

where

$$\begin{aligned}\Delta_1 &= \{x : \alpha_x^{(r_{x_1},u_{x_1})} \,|\, x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)\}\\ \Delta_2 &= \{x : \alpha_x^{(r_{x_2},u_{x_2})} \,|\, x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)\}\\ C_1 &= \{\Delta_1(x) \preceq \Gamma_1(x) : x \in \mathrm{dom}(\Delta_1)\}\\ C_2 &= \{\Delta_2(x) \preceq \Gamma_2(x) : x \in \mathrm{dom}(\Delta_2)\},\end{aligned}$$

with $\beta_x, r_x, u_x, \alpha_x, r_{x_1}, r_{x_2}, u_{x_1}$, *and* u_{x_2} *fresh for each* x.

Operations $+, \sqcup$, and $\times$ are only defined for types that differ at most in their outermost uses. Nevertheless, an operation between types can still be defined for some common subtype of both. That is the role that type environments Δ_1 and Δ_2 play in the above definition—they offer a common subtype for names that belong to Γ_1 and Γ_2. Notice that, despite the fact that α_x is fresh, it is assigned to $\Delta_1(x)$ and $\Delta_2(x)$, however with distinct use variables.

Definition 31. $\Gamma_1 \sqcup \Gamma_2$ *is the pair* $\Gamma; C$ *where*

$$\begin{aligned}\Gamma &= \{x : \alpha_x^{(r_x,u_x)} \,|\, x \in \mathrm{dom}(\Gamma_1) \cup \mathrm{dom}(\Gamma_2)\}\\ C &= \{\Gamma(x) \preceq \Gamma_i(x) \,|\, x \in \mathrm{dom}(\Gamma_i)\},\end{aligned}$$

with α_x, r_x *and* u_x *fresh for each* x.

Definition 32. $\kappa \otimes \Gamma$ *is the pair* $\Delta; C$ *where*

$$\begin{aligned}\Delta &= \{x : \alpha_x^{(r_x,u_x)} \,|\, x \in \mathrm{dom}(\Gamma)\}\\ C &= \{\Delta(x) \preceq \kappa \cdot \Gamma(x) \,|\, x \in \mathrm{dom}(\Gamma)\},\end{aligned}$$

with α_x, r_x *and* u_x *fresh for each* x.

The following proposition establishes the relation between the operators $\oplus, \sqcup, \otimes$ and the satisfaction of subtype constraints constructed from operators $+, \sqcup$ and $\times$, respectively.

Proposition 33.

1. *If* $\Gamma; C = \Gamma_1 \oplus \Gamma_2$, *then* $C \models (\Gamma \preceq \Gamma_1 + \Gamma_2)$.
2. *If* $\Gamma; C = \Gamma_1 \sqcup \Gamma_2$, *then* $C \models (\Gamma \preceq \Gamma_1 \sqcup \Gamma_2)$.
3. *If* $\Gamma; C = \kappa \otimes \Gamma_1$, *then* $C \models (\kappa \times \Gamma_1)$.

Proof. See Igarashi and Kobayashi [IK00]. □

We now present a type reconstruction algorithm (LTR—Linear Type Reconstruction) that can infer principal typings from process expressions. For some process P the algorithm computes a pair $\Gamma; C$ where Γ is a type environment and C represents a set of restrictions on subtyping expressions. The algorithm mimics the syntax-directed rules, using the operators $\oplus, \otimes, \sqcup$ to explicitly determine the subtype constraints.

In what follows, we use the untyped version of the syntax for processes (defined in section 2). Since our purpose is to determine the type of names from process expressions it does not make much sense to explicitly indicate the type of the bound name in a $\mathsf{new}\ x : \rho\ P$ process. In fact, we can infer ρ from P. Therefore, for type reconstruction purposes, we write $\mathsf{new}\, x\ P$ instead of the usual $\mathsf{new}\, x : \rho\ P$.

$$\mathrm{LTR}(a\,!\,l_i[\tilde{v}]) = a : \{l_1 : t_1^{(r_1,u_1)}, \ldots, l_n : t_n^{(r_n,u_n)}\}^{(0,1)} \oplus \tilde{v} : t_i^{(r_i,u_i)}$$

$$\begin{aligned}\mathrm{LTR}(a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}) &= \Gamma; C \cup C_1 \cup \ldots \cup C_n \cup C' \\ \text{where } \Gamma_1; C_1 &= \mathrm{LTR}(P_1) \\ &\vdots \\ \Gamma_n; C_n &= \mathrm{LTR}(P_n) \\ \Pi; C' &= (\Gamma_1 \setminus \tilde{x}_1) \sqcup \ldots \sqcup (\Gamma_n \setminus \tilde{x}_n) \\ \Gamma; C &= \Pi \oplus a : \{l_1 : \Gamma_1(\tilde{x}_1), \ldots, l_n : \Gamma_n(\tilde{x}_n)\}^{(1,0)}\end{aligned}$$

$$\begin{aligned}\mathrm{LTR}(\mathsf{new}\ x\ P) &= (\text{if } x \in \mathrm{dom}(\Gamma) \text{ then } \Gamma \setminus x \text{ else } \Gamma); C \\ \text{where } (\Gamma, C) &= \mathrm{LTR}(P)\end{aligned}$$

$$\begin{aligned}\mathrm{LTR}(P \mid Q) &= \Gamma; C \cup C_1 \cup C_2 \\ \text{where } \Gamma_1; C_1 &= \mathrm{LTR}(P) \\ \Gamma_2; C_2 &= \mathrm{LTR}(Q) \\ \Gamma; C &= \Gamma_1 \oplus \Gamma_2\end{aligned}$$

$$\mathrm{LTR}(X[\tilde{v}]) = (\{X : t_1^{(r_1,u_1)} \ldots t_n^{(r_n,u_n)}, \tilde{v} : t'^{(r',u')}\}, t'^{(r',u')} \preceq t_i^{(r_i,u_i)}).$$

$$\mathrm{LTR}(\mathsf{0}) = (\emptyset, \emptyset).$$

$$\mathrm{LTR}(\mathsf{def}\ X_1\,(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\ \mathsf{in}\ Q) =$$

$$\Gamma; C_1 \cup \cdots \cup C_n \cup C' \cup C'' \cup C'''$$

where $(\Gamma_1, C_1) = \mathrm{LTR}(P_1)$

$\vdots$

$\Gamma_n; C_n = \mathrm{LTR}(P_n)$

$\Delta; C' = \mathrm{LTR}(Q)$

$\Pi; C'' = \bigoplus_i (\mathcal{U}(X_i, D, Q) \otimes (\Gamma_i \setminus \tilde{x}_n))$

$\Gamma; C''' = \Delta \oplus \Pi$

Theorem 34 (Correctness of the algorithm).

1. *If* $\Gamma; C \vdash P$, *then* $\mathrm{LTR}(P)$ *outputs the principal typing of* P.
2. *If* $\mathrm{LTR}(P)$ *outputs* $\Gamma; C$, *then* $\Gamma; C$ *is a principal typing of* P.

Proof. See appendix A. □

The following example illustrates the application of the LTR algorithm.

Example 35. *Consider the process expression*

$$P \equiv a\,!\,l_1[v] \,|\, a\,?\,\{l_1\,(x) = x\,!\,l[y]\}.$$

$\mathrm{LTR}(a\,!\,l_1[v] \,|\, a\,?\,\{l_1\,(x) = x\,!\,l[y]\}) = \Gamma; C \cup C_1 \cup C_2$
where $\Gamma_1; C_1 = \mathrm{LTR}(a\,!\,l_1[v])$
$\Gamma_2; C_2 = \mathrm{LTR}(a\,?\,\{l_1 = x\,!\,l[y]\})$
$\Gamma; C = \Gamma_1 \oplus \Gamma_2$.

Now we compute $\Gamma_1; C_1$.
$\mathrm{LTR}(a\,!\,l_1[v]) = a : \{l_1 : t_1^{(r_1,u_1)}\}^{(0,1)} \oplus v : t_1^{(r_1,u_1)}$.
Then $\Gamma_1 = \{a : t_2^{(r_2,u_2)}, v : t_3^{(r_3,u_3)}\}$ *and*
$C_1 = \{t_2^{(r_2,u_2)} \preceq \{l_1 : t_1^{(r_1,u_1)}\}^{(0,1)}, t_3^{(r_3,u_3)} \preceq t_1^{(r_1,u_1)}\})$.

To compute $\Gamma_2; C_2$, *we let*
$\mathrm{LTR}(a\,?\,\{l_1\,(x) = x\,!\,l[y]\}) = \Pi; C' \cup C'_1$
where $\Delta; C'_1 = \mathrm{LTR}(x\,!\,l[y])$
$\Pi; C' = (\Delta \setminus x) \oplus a : \{l : \Delta(x)\}^{(1,0)}$

It is easy to compute
$\Delta = \{x : t_5^{(r_5,u_5)}, y : t_6^{(r_6,u_6)}\}, \qquad \Pi = \{y : t_7^{(r_7,u_7)}, a : t_8^{(r_8,u_8)}\},$
$C'_1 = \{t_5^{(r_5,u_5)} \preceq \{l : t_4^{(r_4,u_4)}\}^{(0,1)}, t_6^{(r_6,u_6)} \preceq t_4^{(r_4,u_4)}\},$
$C' = \{t_7^{(r_7,u_7)} \preceq t_6^{(r_6,u_6)}, t_8^{(r_8,u_8)} \preceq \{l_1 : t_5^{(r_5,u_5)}\}^{(1,0)}\}.$

Which means that $\Gamma_2 = \Pi$ *and* $C_2 = C' + C'_1$.

Hence, the algorithm output

$$\begin{aligned}\Gamma = \Gamma_1 + \Pi = \{&a : t_{10}^{(r_{10},u_{10})}, y : t_{11}^{(r_{11},u_{11})}, v : t_{12}^{(r_{12},u_{12})}\},\\ C = \{&t_9^{(r_9,u_9)} \preceq t_2^{(r_2,u_2)}, t_9^{(r'_9,u'_9)} \preceq t_8^{(r_8,u_8)}\} \cup\\ &\{t_{10}^{(r_{10},u_{10})} \preceq t_9^{(r_9,u_9)} + t_9^{\prime(r'_9,u'_9)}, t_{11}^{(r_{11},u_{11})} \preceq t_7^{(r_7,u_7)}\} \cup\\ &\{t_{12}^{(r_{12},u_{12})} \preceq t_3^{(r_3,u_3)}\} \cup C_1 \cup C_2.\end{aligned}$$

Resolving the subtype constraint set C, we get

$$\begin{aligned}&a : \{l_1 : \{l : t_{11}^{(0,0)}\}^{(0,1)}\}^{(1,1)}, \quad v : \{t_{11}^{(0,0)}\}^{(0,1)},\\ &x : \{t_{11}^{(0,0)}\}^{(0,1)}, \qquad\qquad\quad y : t_{11}^{(0,0)}.\end{aligned}$$

6 The usage of a process variable

We now present an algorithm to compute the number of times that a process variable is called in a def process.

Notice that the algorithm has to deal with recursive calls to procedures (possibly defined using mutually recursive equations) and with free names that may occur in each definition body.

Our approach is to interpret procedure calls as a graph that models the dependencies between each process variable. The number of times $(0, 1, \omega)$ that a certain variable X is called in a given process P is then given by the number of paths starting on every $Y \in \mathrm{fv}(P)$ and ending in X. We formalise these concepts in what follows.

Definition 36 (Follows or is reachable). *Let Y be a process variable and P be a process* def $X_1\,(\tilde{x}_1) = P_1$ and $\ldots$ and $X_n\,(\tilde{x}_n) = P_n$ in Q. *We say that Y* follows directly from *X_i, denoted by $X_i \to_1 Y$, if $P_i \equiv$* new $x_1 : \rho_1$ $\ldots$ new $x_n : \rho_n$ def D_1 in $\ldots$ def D_m in $Y[\tilde{v}] \mid R$, *for some process R and $n \geq 0$, $m \geq 0$.*

The relation $\to$ is the reflexive-transitive closure of $\to_1$. When $X \to Y$ we say that Y follows from *or* is reachable from *X.*

The algorithm to determine if two nodes are connected in a direct-graph is well-known from graph theory and can be found easily in graph theory literature (for instance [AHU74]). Next we describe the construction of the graph of call dependencies.

Definition 37 (Call dependency graph). *The* graph of call dependencies *of a definition $D \equiv X_1 = (\tilde{x}_1)\,P_1$* and $\cdots$ and *$X_n = (\tilde{x}_n)\,P_n$, denoted by, $\mathcal{D}(D)$ is a direct-graph constructed as follows:*

1. *Insert a node for each process variable in D (bound or free).*
2. *Insert an edge from X_i to each Y, if $X_i \to_1 Y$ holds.*

The recursive function, $\mathcal{U}$, computes the number of times that a process variable X_i is called in $P \equiv \mathsf{def}\ D\ \mathsf{in}\ Q \equiv \mathsf{def}\ X_1(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n(\tilde{x}_n) = P_n\ \mathsf{in}\ Q$.

$$\mathcal{U}(X_i, D, Q) \stackrel{\text{def}}{=} \mathcal{U}'(X_i, D, Q, \emptyset).$$

The auxiliary function $\mathcal{U}'$ maintains a set of process variables V to track those already visited. This is necessary to avoid infinite recursion as $\mathcal{U}'$ descends the process tree structure.

We assume, without lost of generality, that all bound process variables X_j are different form X'_j.

1. $\mathcal{U}'(X_i, D, \mathsf{0}, V) = 0$
2. $\mathcal{U}'(X_i, D, a\,!\,l_i[\tilde{v}], V) = 0$
3. $\mathcal{U}'(X_i, D, P \mid Q, V) = \mathcal{U}'(X_i, D, P, V) + \mathcal{U}'(X_i, D, Q, V)$
4. $\mathcal{U}'(X_i, D, a\,?\,\{l_1(\tilde{x}_1) = P_1, \ldots, l_n(\tilde{x}_n) = P_n\}, V) = \bigsqcup_j \mathcal{U}'(X_i, D, P_j, V)$
5. $\mathcal{U}'(X_i, D, \mathsf{new}\ x : \rho\ P, V) = \mathcal{U}'(X_i, D, P, V)$
6. $\mathcal{U}'(X_i, D, Y[\tilde{v}], V) = 1$, if $Y \notin \mathrm{bv}(D)$ and $X_i = Y$
   $\mathcal{U}'(X_i, D, Y[\tilde{v}], V) = 0$, if $Y \notin \mathrm{bv}(D)$ and $X_i \neq Y$ or $Y \in \mathrm{bv}(D), Y \in V$, and $Y \not\rightarrow X_i$
   $\mathcal{U}'(X_i, D, Y[\tilde{v}], V) = \omega$, if $Y_j \in \mathrm{bv}(D), Y \in V$, and $Y \rightarrow X_i$
   $\mathcal{U}'(X_i, D, Y_j[\tilde{v}], V) = 1 + \mathcal{U}'(X_i, D, P_j, V \cup \{Y_j\}))$, if $Y_j \in \mathrm{bv}(D), Y_j \notin V$, and $X_i = Y_j$
   $\mathcal{U}'(X_i, D, Y_j[\tilde{v}], V) = \mathcal{U}'(X_i, D, P_j, V \cup \{Y_j\}))$, if $Y_j \in \mathrm{bv}(D), Y_j \notin V$, and $X_i \neq Y_j$
7. $\mathcal{U}'(X_i, D, \mathsf{def}\ D'\ \mathsf{in}\ Q', V) = \mathcal{U}'(X_i, D', Q', V) + \sum_j \{\mathcal{U}'(X_j, D', Q', V) \times \mathcal{U}'(X_i, D, P_j, V)\}$

where D is $X_1(\tilde{x}_1) = P_1$ and $\ldots$ and $X_n(\tilde{x}_n) = P_n$ and D' is $X'_1(\tilde{x}_1) = P'_1$ and $\ldots$ and $X'_n(\tilde{x}_n) = P'_n$.

The function $\mathcal{U}'$ counts the number of times that X_i is called in $\mathsf{def}\ D\ \mathsf{in}\ Q$, by analysing the number of calls to X_i in Q. If Q is an atomic process (*i.e.* inaction or message process) the count is trivial. Otherwise $\mathcal{U}'$ must descend the process structure and analyse each subprocess of Q. The cases that worth mention are:

1. In case 4, we compute the least upper bound use of X_i (not the sum, as might be expected). This is so because only one method is selected in reduction. Thus, is it enough to consider the least upper bound use of X_i in all procedure definitions.
2. First and second clauses of case 6 assert that we cannot descend the process structure of a procedure that is not defined in the def process that we are analysing.

The reachability tests performed at clauses 2 and 3 of rule 6 are necessary when a variable was already visited ($Y \in V$). This means that there is a cycle starting in Y, since Y is the first process variable that belongs to V. Thus, if X_i is part of that cycle its use is obviously ω, otherwise is 0.

3. Case 7 describes the number of calls to X_i from a nested def process. We must consider two cases: when X_i is called directly from def D' in Q', that is, a call to X_i is explicitly mentioned in the nested def. That is what $\mathcal{U}'(X_i, D', Q', V)$ counts. But, X_i can also be called indirectly by some X_j as well. Then, we have to count the usage of each X_j in the inner def process and, after that, find if X_i is called from that particular X_j. Thus, the total number of calls to X_i is the product of the calls to X_j in the inner def by the calls to X_i in the process bound to X_j.

Our last result states that $\mathcal{U}$ defined as above is a call counting function.

Theorem 38. *Function $\mathcal{U}$ is a call-counting function.*

Proof. See appendix A. □

To illustrate $\mathcal{U}$ consider the example 14 on page 10. The result of $\mathcal{U}(X_i, D, Q)$ for $1 \leq i \leq 5$ is, respectively, 1, 1, ω, ω, and 0. These results are more accurate than $\omega = W(X_i, D, Q)$, for $1 \leq i \leq 5$.

7 Conclusions

In this report we have shown how to extend TyCO type system [Vas99] to incorporate linear channels. We follow Pierce, Sangiorgi, Kobayashi, Turner, and Igarashi [PS96, KPT99, IK00], and specify a type system that distinguishes input, output, and input/output channels, counting the number of times that channels are used in one way or another. We enunciated the properties that a function must satisfy in order to be a call-counting function and have devised a function $\mathcal{U}$ that computes accurately such channel usage. This opens the possibility of compiler optimisations suggesting that linear channels need not be implemented using queues, and that the channels memory may be deallocated after communication occurs.

We also devised an algorithm (adapted from [IK00]) to perform type reconstruction which means that, from a practical point of view, programmers need not give any kind of information related to channels usage. All usage information is recovered from process expressions by type inference.

It remains for further study the impact of possible optimisations in the TyCO compiler and the applicability of source code transformation techniques to optimise programs, knowing information about channels usage.

References

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Series in Computer Science and Information Processing. Addison-Wesley, 1974.

[IK00] Atsushi Igarashi and Naoki Kobayashi. Type reconstruction for linear pi-calculus with i/o subtyping. *Information and Computation*, 161:1–44, 2000.

[KPT99] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, 1999.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part i and ii. *Information and Computation*, 100(1):1–77, 1992.

[PS96] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Journal of Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

[Vas99] Vasco T. Vasconcelos. Processes, functions, datatypes. *Theory and Practice of Object Systems*, 5(2):97–110, 1999.

[VH92] Vasco T. Vasconcelos and Kohei Honda. Principal typing-schemes in a polyadic π-calculus. CS 92–004, Keio University, November 1992.

A Proofs

This section presents proofs for the theorems stated in the report; it includes other definitions and results needed by such proofs.

On the proofs that follow, we use (implicitly) the variable convention whenever necessary and take all the names in P to be different from each other (by α-convert bound names).

A.1 Proofs for section 4

Essentially subject-reduction. In this section we redefine the meaning for the symbol $\preceq$.

Definition 39. $\Gamma' \preceq \Gamma$, *if* $\forall x \in \mathrm{dom}(\Gamma)$, *then* $x \in \mathrm{dom}(\Gamma')$ *and* $\Gamma'(x) \preceq \Gamma(x)$.

The next lemma states that, if there is a Γ, such that $\Gamma \vdash P$, then there exists Δ where $\Delta \vdash P$ and $\Gamma \preceq \Delta$.

Lemma 40. *Suppose that* $\Gamma \vdash P$.

1. *If* $P \equiv a\,!\,l_i[\tilde{v}]$, *then there exists* $\rho = \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(0,1)}$ *and* $\tilde{\tau}$ *such that* $\Gamma \preceq a : \rho + \tilde{v} : \tilde{\tau}$.
2. *If* $P \equiv a\,?\,\{M\}$, *then there exists* Δ *such that* $\Delta \vdash M : \alpha$ *and* $\Gamma \preceq \Delta + a : \alpha^{(1,0)}$.
3. *If* $P \equiv P_1 \mid P_2$, *then there exists* Γ_1 *and* Γ_2 *such that* $\Gamma_1 \vdash P_1$, $\Gamma_2 \vdash P_2$ *and* $\Gamma \preceq \Gamma_1 + \Gamma_2$.
4. *If* $P \equiv \mathsf{new}\ x : \rho\ R$, *then* $\Gamma \uplus x : \rho \vdash R$.
5. *If* $P \equiv X[\tilde{v}]$, *then there exists* $\tilde{\tau}$ *such that* $\Gamma \preceq \tilde{v} : \tilde{\tau}$.
6. *If* $P \equiv \mathsf{def}\ D\ \mathsf{in}\ Q$, *then there exists* Δ *such that* $\Delta \vdash P$ *and* $\Gamma \preceq \Delta$.
7. *If* $P \equiv \mathsf{0}$, *then there exists* Δ *such that* $\Delta \vdash \mathsf{0}$ *and* $\Gamma \preceq \Delta$.

Proof.

1. If $P \equiv a\,!\,l_i[\tilde{v}]$ and $\Gamma \vdash P$, then

   $$\Gamma(a) = \alpha_1^{(\kappa_1,\kappa_2)} = \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(\kappa_1,\kappa_2)} \qquad \text{and } \Gamma(\tilde{v}) = \tilde{\tau}.$$

   Since $\Gamma \vdash P$, then τ must be a subtype of ρ_i and therefore, $\alpha_1^{(0,1)} \preceq \alpha_2^{(0,1)} = \{l_1 : \tilde{\rho}_1, \cdots, l_i : \tilde{\tau}, \cdots, l_n : \tilde{\rho}_n\}^{(0,1)}$. Hence,

   $$\textsc{Sub}\ \frac{\textsc{Msg}\ \dfrac{}{a : \alpha_2^{(0,1)} + \tilde{v} : \tilde{\tau} \vdash P} \qquad \alpha_1^{(0,1)} \preceq \alpha_2^{(0,1)}}{a : \alpha_1^{(0,1)} + \tilde{v} : \tilde{\tau} \vdash P},$$

and thus,

$$\textsc{Sub}\ \frac{a : \alpha_1^{(0,1)} + \tilde{v} : \tilde{\tau} \vdash P \qquad \alpha_1^{(\kappa_1,\kappa_2)} \preceq \alpha_1^{(0,1)}}{a : \alpha_1^{(\kappa_1,\kappa_2)} + \tilde{v} : \tilde{\tau} \vdash P}.$$

For all names $x \in \mathrm{dom}(\Gamma)$, such that $x \neq a$ and $x \neq v_i$ one can apply Weak (for names and process variables), and conclude that $\Gamma \vdash P$, with $\Gamma \preceq a : \alpha_2^{(0,1)} + \tilde{v} : \tilde{\tau}$.

2. if $P \equiv a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}$ and $\Gamma \vdash P$, then

$$\textsc{Obj}\ \frac{\textsc{Meth}\ \dfrac{\Delta \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \cdots \Delta \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n}{\Delta \vdash l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n : \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}}}{\Delta + a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(1,0)} \vdash a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}}$$

Since $\Gamma \vdash P$, $\Gamma(a) \preceq \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(0,1)}$,

$$\textsc{Sub}\ \frac{\Delta + a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(0,1)} \qquad \Gamma(a) \preceq \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(0,1)}}{\Delta + a : \Gamma(a) \vdash a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}}.$$

For each $x \in \mathrm{dom}(\Gamma)$ such that $x \notin \mathrm{dom}(\Delta)$ we apply the Weak rule. Finally one gets $\Gamma \preceq \Delta + a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(1,0)}$.

3. If $P \equiv P_1 \,|\, P_2$ and $\Gamma \vdash P_1 \,|\, P_2$, then, by Par rule, there exists Δ_1 and Δ_2, such that

$$\textsc{Par}\ \frac{\Delta_1 \vdash P_1 \qquad \Delta_2 \vdash P_2}{\Gamma \vdash P_1 \,|\, P_2}, \Gamma = \Delta_1 + \Delta_2.$$

By induction hypothesis, if $\Delta_i \vdash P_i$, then there exists Γ_i such that

$$\begin{array}{rcl} \Gamma_1 \vdash P_1 & \text{and} & \Delta_1 \preceq \Gamma_1 \\ \text{and}\quad \Gamma_2 \vdash P_2 & \text{and} & \Delta_2 \preceq \Gamma_2 \end{array}.$$

Then, from Par we have

$$\textsc{Par}\ \frac{\Delta_1 \preceq \Gamma_1 \vdash P_1 \qquad \Delta_2 \preceq \Gamma_2 \vdash P_2}{\Gamma = \Delta_1 + \Delta_2 \preceq \Gamma_1 + \Gamma_2 \vdash P_1 \,|\, P_2}$$

and hence there exists Γ_1 and Γ_2 such that $\Gamma \preceq \Gamma_1 + \Gamma_2$.

4. This case follows straightforwardly from the application of the Res rule. Nevertheless, one could show that, since $\Gamma \vdash \mathsf{new}\ x : \rho\ P$, there exists $\Delta \vdash \mathsf{new}\ x : \rho\ P$ and $\Gamma \preceq \Delta$. For that, one only need to consider that

$$\textsc{Res}\ \frac{\Gamma \uplus x : \rho \vdash P}{\Gamma \vdash \mathsf{new}\ x : \rho\ P}$$

then, by hypothesis, there exists $\Delta \uplus x : \rho$ such that

$$\Delta \uplus x : \rho \vdash P \qquad \text{and} \qquad \Gamma \cdot x : \rho \preceq \Delta \uplus x : \rho.$$

Hence, Δ verifies

$$\Delta \vdash \mathsf{new}\ x : \rho\ P \qquad \text{and} \qquad \Gamma \preceq \Delta.$$

5. By hypothesis $\Gamma \vdash X[\tilde{v}]$. Then one has

$$\Delta \uplus X : \tilde{\rho} \uplus \tilde{v} : \tilde{\sigma} \vdash X[\tilde{v}].$$

By App, we know that

$$X : \tilde{\rho} \uplus \tilde{v} : \tilde{\rho} \vdash X[\tilde{v}]$$

and, since $\Gamma \vdash X[\tilde{v}]$, then $\tilde{\sigma} \preceq \tilde{\rho}$. Hence, by Sub rule,

$$X : \tilde{\rho} \uplus \tilde{v} : \tilde{\sigma} \vdash X[\tilde{v}].$$

By successive applications of Weak, we conclude the case.

6. By hypothesis and def, there exists Λ and $\Gamma_1, \cdots, \Gamma_n$ such that:

$$\textsc{Def}\ \frac{\begin{array}{c}\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \\ \vdots \\ \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n \\ \Lambda \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \vdash Q\end{array}}{\Gamma \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\ \mathsf{in}\ Q},$$

with $\Gamma = \sum_i \mathcal{U}(X_i, D, Q) \cdot \Gamma_i + \Lambda$.
By induction hypothesis, there exists $\Delta_1, \cdots, \Delta_n$ and Π such that

$$\begin{array}{l}\Delta_1 \vdash P_1 \text{ and } \Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_1 : \tilde{\rho}_1 \preceq \Delta_1 \\ \qquad\qquad \vdots \\ \Delta_n \vdash P_n \text{ and } \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \preceq \Delta_n \\ \text{and } \Pi \vdash Q \text{ and } \Lambda \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \preceq \Pi\end{array}.$$

Thus, applying again Def, we get

$$\textsc{Def}\ \frac{\begin{array}{c}\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus; \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_1 : \tilde{\rho}_1 \preceq \Delta_1 \vdash P_1 \\ \vdots \\ \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \preceq \Delta_n \vdash P_n \\ \Lambda \cdot X_1 : \tilde{\rho}_1 \uplus \ \cdots \ \uplus X_n : \tilde{\rho}_n \preceq \Pi \vdash Q\end{array}}{\begin{array}{r}\Gamma = \sum_i \mathcal{U}(X_i, D, Q) \cdot \Gamma_i + \Lambda \preceq \sum_i \mathcal{U}(X_i, D, Q) \cdot \Delta_i + \Pi \\ \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\ \mathsf{in}\ Q\end{array}}.$$

Notice that the function U yields the same results for the two type environments, since process expressions remain the same. Hence, we've proved that $\Delta = \sum_i \mathcal{U}(X_i, D, Q) \cdot \Delta_i + \Pi$ exists, $\Delta \vdash P$, and that $\Gamma \preceq \Delta$.

7. This one is straightforward because $\emptyset \vdash \mathsf{0}$ and $\Gamma \preceq \emptyset$ for any Γ.

□

Proposition 41. *Let* $\tau = \alpha_1^{(\kappa_1,\kappa_2)}$ *and* $\rho = \alpha_2^{(\mu_1,\mu_2)}$ *such that* $\tau + \rho$ *is defined. Then,* $\tau + \rho \preceq \tau$ *and* $\tau + \rho \preceq \rho$.

Proof. In order to $\tau + \rho$ be defined, τ and ρ must differ only on their outermost uses (see definition 2, page 4). Therefore α_1 and α_2 denote the same type. Thus, $\tau + \rho = \alpha_1^{(\kappa_1,\kappa_2)} + \alpha_1^{(\mu_1,\mu_2)} = \alpha_1^{(\kappa_1+\mu_1,\kappa_2+\mu_2)} \preceq \alpha_1^{(\kappa_1,\kappa_2)}$, since $\kappa_1 + \mu_1 \geq \kappa_1$ and $\kappa_2 + \mu_2 \geq \kappa_2$ holds for every use expression. The same argument can be followed for $\tau + \rho \preceq \rho$. □

The following lemma is the core of the *substitution lemma* and let us reason about substitutions. Typically the substitution lemma is only concerned with name replacement and does not refer to types. Nevertheless, as we have to keep track of names usage, we must take special care when performing these substitutions.

As an outline of the theorem, consider that we want to replace y for z in P. Then, the usage of z must reflect also the usage of y (since z replaces y in P). Thus, we need to sum together the usages of y and z in order to correctly typify P. As a consequence, we have to reason about types also as well as to establish conditions that define when it is possible to sum the types ρ (of y) and τ (of z). Hence, in order to z replace y, we require that it must exists subtypes of ρ and τ where the sum can be defined. Notice that albeit $\rho + \tau$ may not be defined, $\rho' + \tau'$ can be defined for some $\rho' \preceq \rho$ and $\tau' \preceq \tau$. That is what the first and third conditions of the lemma specify.

When performing the substitution $\tilde{y}$ for $\tilde{z}$, the names of the $\tilde{z}$ sequence need not be all distinct. Then, we must guarantee that is also possible to sum the identical z_i because its usage must be considered together. That is why we require the second condition.

Lemma 42. *If* $\Gamma \uplus z_1 : \tau_{z_1} + \cdots + z_n : \tau_{z_n} \uplus \tilde{y} : \tilde{\tau}_y \vdash P$ *and if there exists* $\tilde{\tau}'_z$ *and* $\tilde{\tau}'_y$ *such that*

1. $\tau'_{z_i} \preceq \tau_{z_i}$ *and* $\tau'_{y_i} \preceq \tau_{y_i}$,
2. $z_1 : \tilde{\tau}'_{z_1} + \cdots + z_n : \tilde{\tau}'_{z_n}$ *is defined,*
3. $\tau'_{z_i} + \tau'_{y_i}$ *is defined,*

then $\Gamma \uplus z_1 : \left(\tau'_{z_1} + \tau'_{y_1}\right) + \cdots + z_n : \left(\tau'_{z_n} + \tau'_{y_n}\right) \vdash \{\tilde{z}/\tilde{y}\}\, P$.

Proof. By induction on the structure of the derivation of the typing of P. We analyse the last typing rule applied.

1. Case MSG. Then $P \equiv a\,!\,l_i[\tilde{v}]$, and by hypothesis, $\Gamma \vdash P$. Thus, from lemma 40, $\Gamma \preceq a : \{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(0,1)} + \tilde{v} : \tilde{\sigma}_i$.

We have several cases to analyse depending on the substitutions to perform, but all these cases are treated similarly. Hence, we consider only the substitution of a by some x. Therefore, we have $\rho = \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(\kappa_1,\kappa_2)}$ and $\tau = \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(\kappa_3,\kappa_4)}$, with $\rho \preceq \{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(0,1)}$, such that

$$\Gamma = \Delta \uplus a : \rho \uplus x : \tau \vdash a\,!\,l_i[\tilde{v}].$$

By hypothesis, there exists ρ' and τ' such that $\rho' \preceq \rho$, $\tau' \preceq \tau$, and $\tau' + \rho'$ is defined.

We want to prove that, $\Gamma = \Delta \uplus x : (\rho' + \tau') \vdash \{x/a\}P$.

Since $\{x/a\}\ P \equiv x\,!\,l_i[\tilde{v}]$, by Msg

$$\Pi = x : \{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(0,1)} + \tilde{v} : \tilde{\sigma}_i \vdash x\,!\,l_i[\tilde{v}],$$

and by Sub,

$$\textsc{Sub}\ \frac{\Pi \vdash x\,!\,l_i[\tilde{v}] \qquad \rho' \preceq \rho \preceq \{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(0,1)}}{x : \rho' + \tilde{v} : \tilde{\sigma}_i \vdash x\,!\,l_i[\tilde{v}]}.$$

Therefore, using proposition 41,

$$\textsc{Sub}\ \frac{x : \rho' + \tilde{v} : \tilde{\sigma}_i \vdash x\,!\,l_i[\tilde{v}] \qquad \rho' + \tau' \preceq \rho'}{x : \rho' + \tau' + \tilde{v} : \tilde{\sigma}_i \vdash x\,!\,l_i[\tilde{v}]}.$$

By successive applications of Sub and Weak, we conclude that

$$\Gamma = \Delta \uplus x : \rho' + \tau' \vdash x\,!\,l_i[\tilde{v}],$$

which proves the case.

2. Case Nil. Then $P \equiv \mathsf{0}$, and, as the *inaction* process does not have names, any substitution have no effect on it. By Nil and successive applications of Weak we can prove that any type environment Γ typifies $\mathsf{0}$, particularly the environment in the theorem thesis.

3. Case App. Then $P \equiv X[\tilde{v}]$, and by hypothesis, $\Gamma \vdash P$. Thus, from lemma 40, $\Gamma \preceq \tilde{v} : \tilde{\sigma}$.

   The only substitution that makes sense to analyse is when some (or all) v_i are substituted by x_i. Therefore, we have $\tilde{\rho} = \rho_1 \cdots \rho_n$ and $\tilde{\tau} = \tau_1 \cdots \tau_n$, with $\tilde{\rho} \preceq \tilde{\sigma}$, such that

   $$\Gamma = \Delta \uplus \tilde{v} : \tilde{\rho} \uplus \tilde{x} : \tilde{\tau} \vdash X[\tilde{v}].$$

   By hypothesis, there exists $\tilde{\rho}'$ and $\tilde{\tau}'$ such that $\tilde{\rho}' \preceq \tilde{\rho}$, $\tilde{\tau}' \preceq \tilde{\tau}$, $x_1 : \tau'_1 + \cdots + x_n : \tau'_n$ is defined, and $\tau'_i + \rho'_i$ is defined (for

$1 \leq i \leq n$). We want to prove that, $\Gamma = \Delta \uplus x_1 : (\rho'_1 + \tau'_1) + \cdots + x_n : (\rho'_n + \tau'_n) \vdash \{\tilde{x}/\tilde{v}\}P$.
Since $\{\tilde{x}/\tilde{v}\}P \equiv X[\tilde{x}]$, by App

$$\Pi = X : \sigma_1 \cdots \sigma_n \uplus \tilde{x} : \sigma_1 \cdots \sigma_n \vdash X[\tilde{x}],$$

and by Sub,

$$\textsc{Sub}\ \frac{\Pi \vdash X[\tilde{x}] \qquad \tilde{\rho}' \preceq \tilde{\rho} \preceq \tilde{\sigma}}{X : \tilde{\sigma} \uplus \tilde{x} : \tilde{\rho}' \vdash X[\tilde{x}]}.$$

Therefore, using proposition 41,

$$\textsc{Sub}\ \frac{X : \tilde{\sigma} \uplus \tilde{x} : \tilde{\rho}' \vdash X[\tilde{x}] \qquad \rho' + \tau' \preceq \rho'}{X : \tilde{\sigma} \uplus \tilde{x} : (\tilde{\rho}' + \tilde{\tau}') \vdash X[\tilde{x}]}.$$

By successive applications of Sub and Weak, we conclude that

$$\Gamma = \Delta \uplus x_1 : (\rho'_1 + \tau'_1) + \cdots + x_n : (\rho'_n + \tau'_n) \vdash X[\tilde{x}],$$

which proves the case.

4. Case Obj. Then $P \equiv a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}$. If $\Gamma \vdash P$, then

$$\textsc{Obj}\ \frac{\textsc{Meth}\ \dfrac{\Delta \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \quad \cdots \quad \Delta \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n}{\Delta \vdash l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n : \alpha}}{\Delta + \alpha^{(1,0)} \vdash a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}},$$

with $\alpha = \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}$ and $\Gamma \preceq \Delta + a : \alpha^{(1,0)}$, by lemma 40.

We analyse next the substitution of z for a (that can or not belong to $\mathrm{fn}(P)$). The remaining cases, where the substitution occurs inside a P_i, follows from induction hypothesis and Sub rule.

Let

$$\Gamma \uplus z : \tau_z, a : \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(1,0)} \vdash P \tag{1}$$

and

$$\begin{array}{rcl} \tau'_z & \preceq & \tau_z \\ \{l_1 : \tilde{\rho}'_1, \cdots, l_n : \tilde{\rho}'_n\}^{(\kappa'_{1a}, \kappa'_{2a})} & \preceq & \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(1,0)} \end{array}$$

such that, $\tau'_z + \{l_1 : \tilde{\rho}'_1, \cdots, l_n : \tilde{\rho}'_n\}^{(\kappa'_{1a}, \kappa'_{2a})}$ is defined.
We want to prove that $\Gamma \uplus z : \tau'_z + \rho'_a \vdash \{z/a\}\ P$.

By hypothesis one know that $\Gamma' \uplus z : \tau'_z + \rho'_a \vdash \{z/a\}\{l_1 = (\tilde{x}_1)\, P_1, \cdots, (\tilde{x}_n)\, P_n\}$. One knows that,

$$\begin{aligned}\{z/a\}(a\,?\,\{l_1 = (\tilde{x}_1)\, P_1, \cdots, l_n = (\tilde{x}_n)\, P_n\}) \\ = z\,?\,\{l_1 = (\tilde{x}_1)\, P_1, \cdots, l_n = (\tilde{x}_n)\, P_n\}\end{aligned}$$

and by hypothesis, using Sub twice

$$\dfrac{\dfrac{\Gamma \uplus z : \tau_z, a : \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(0,1)} \qquad \rho'_a \preceq \rho_n}{\Gamma \uplus z : \tau_z, a : \{l_1 : \tilde{\rho}'_1, \cdots, l_n : \tilde{\rho}'_n\}^{(\kappa'_{1a}, \kappa'_{2a})} \vdash P} \qquad \tau'_z \preceq \tau_z}{\Gamma \uplus z : \tau'_z, a : \{l_1 : \tilde{\rho}'_1, \cdots, l_n : \tilde{\rho}'_n\}^{(\kappa'_{1a}, \kappa'_{2a})} \vdash P}$$

Then, because $\rho' + \tau'_z$ is defined and $\tau'_z : \{l_1 : \tilde{\rho}'_1, \cdots, l_n : \tilde{\rho}'_n\}^{(\kappa'_{1z}, \kappa'_{2z})}$, by Obj we have that

$$\Gamma \uplus z : \tau'_z + \rho'_a \vdash z\,?\,\{l_1 = (\tilde{x}_1)\, P_1, \cdots, l_n = (\tilde{x}_n)\, P_n\}.$$

5. Case Par. Then $P \equiv P_1 \mid P_2$, and by hypothesis we have that,

$$\Gamma = \Delta \uplus z_1 : \sigma_1 + \cdots + z_n : \sigma_n \uplus \tilde{y} : \tilde{\pi} \vdash P_1 \mid P_2,$$

with $\tilde{\sigma}' \preceq \tilde{\sigma}$, $\tilde{\pi}' \preceq \tilde{\pi}$, $\sigma'_i + \pi'_i$ defined (for $1 \leq i \leq n$), and $z_1 : \sigma'_1 + \cdots + z_n : \sigma'_n$ defined.
Since $\Gamma \vdash P_1 \mid P_2$, by lemma 40, there exists Γ_1 and Γ_2, such that $\Gamma \preceq \Gamma_1 + \Gamma_2$. Applying induction hypothesis, we have that

$$\begin{aligned}\Gamma_1 = \Delta_1 \uplus z_1 : \tau_1 + \cdots + z_n : \tau_n \vdash \{\tilde{z}/\tilde{y}\}P_1 \\ \Gamma_2 = \Delta_2 \uplus z_1 : \rho_1 + \cdots + z_n : \rho_n \vdash \{\tilde{z}/\tilde{y}\}P_2\end{aligned}$$

But $\Gamma \preceq \Gamma_1 + \Gamma_2$ means, not only that $z_1 : (\tau_1 + \rho_1) + \cdots + z_n : (\tau_n + \rho_n)$ is defined, but also that $\sigma'_i \preceq \sigma_i \preceq \tau_i + \rho_i$.
Then, the following derivation holds.

$$\textsc{Sub}\dfrac{\textsc{Par}\ \dfrac{\Delta_1 \uplus \tilde{z} : \tilde{\tau} \vdash \{\tilde{z}/\tilde{y}\}P_1 \quad \Delta_2 \uplus \tilde{z} : \tilde{\rho} \vdash \{\tilde{z}/\tilde{y}\}P_2}{\Delta_1 + \Delta_2 \uplus \tilde{z} : (\tilde{\tau} + \tilde{\rho}) \vdash \{\tilde{z}/\tilde{y}\}(P_1 \mid P_2)} \qquad \tilde{\sigma}' \preceq \tilde{\tau} + \tilde{\rho}}{\Delta_1 + \Delta_2 \uplus z_1 : \sigma'_1 + \cdots + z_n : \sigma'_n \vdash \{\tilde{z}/\tilde{y}\}(P_1 \mid P_2)}$$

Therefore, using proposition 41,

$$\textsc{Sub}\ \dfrac{\Delta_1 + \Delta_2 \uplus \tilde{z} : \tilde{\sigma}' \vdash \{\tilde{z}/\tilde{y}\}(P_1 \mid P_2) \qquad \tilde{\sigma}' + \tilde{\pi}' \preceq \tilde{\sigma}'}{\Delta_1 + \Delta_2 \uplus z_1 : (\sigma'_1 + \pi'_1) + \cdots + z_n : (\sigma'_n + \pi'_n) \vdash \{\tilde{z}/\tilde{y}\}(P_1 \mid P_2)}$$

By successive applications of Sub and Weak, we conclude the case.

6. Case RES. Then $P \equiv \mathsf{new}\ x : \rho\ R$, and from hypothesis one has that

$$\Gamma = \Delta \uplus z_1 : \sigma_1 + \cdots + z_n : \sigma_n \uplus \tilde{y} : \tilde{\pi} \vdash \mathsf{new}\ x : \rho\ R,$$

and there exists $\tilde{\sigma}'$ and $\tilde{\pi}'$ such that $\tilde{\sigma}' \preceq \tilde{\sigma}$, $\tilde{\pi}' \preceq \tilde{\pi}$, $\sigma'_i + \pi'_i$ is defined (for $1 \leq i \leq n$), and $z_1 : \sigma'_1 + \cdots + z_n : \sigma'_n$ is defined.

Hence, by lemma 40, $\Gamma \uplus x : \rho \vdash R$. Applying induction hypothesis followed by a RES rule we conclude the case.

□

The next result, a corollary of lemma 42, is used to prove subject reduction.

Corollary 43. *If $\Gamma \uplus \tilde{y} : \tilde{\tau} \vdash P$ and if there exist $\tilde{\tau}' \preceq \tilde{\tau}$ such that $\Gamma + z_1 : \tilde{\tau}'_1 + \cdots + z_n : \tilde{\tau}'_n$ is defined, then $\Gamma + z_1 : \tau'_1 + \cdots + z_n : \tau'_n \vdash \{\tilde{z}/\tilde{y}\}P$.*

Proof. The prove is straightforward. Set the types of $z_i \notin \mathrm{dom}(\Gamma)$ to be ρ_i (ρ_i is the same as τ'_i except that their outer most use pair is $(0,0)$), and let $\tilde{z} : \tilde{\rho}$. Then, the conditions of lemma 42 are met, namely there exists $\tau' \preceq \tau$ (from hypothesis) and $\rho' = \rho \preceq \rho$ ($\preceq$ is reflexive). Moreover, $\Gamma + z_1 : \tilde{\tau}'_1 + \cdots + z_n : \tilde{\tau}'_n$ guaranties that $\tilde{\tau}_i + \tilde{\rho}_i$ and $z_1 : \rho'_1 + \cdots + z_n : \rho'_n$ are defined. Hence, $\Gamma + z_1 : \tau'_1 + \cdots + z_n : \tau'_n \vdash \{\tilde{z}/\tilde{y}\}\, P$. □

The following lemma states that a process remains typified after a name substitution on process expression and type environment. This is the "standard" substitution lemma and only talks about names. Nevertheless, it is a special case of lemma 42.

Lemma 44 (Substitution lemma). *If $\Gamma \vdash P$ and $z \notin \mathrm{fn}(P)$, then $\{z/y\}\Gamma \vdash \{z/y\}P$.*

Proof. We have to consider two cases:

1. If $y \notin \mathrm{dom}(\Gamma)$, then it means that $\{y/z\}\Gamma$ has no effect, neither does $\{z/y\}P$, because y does not occur free in P (or else it would be in Γ). Then, it is trivial that $\{z/y\}\Gamma \vdash \{z/y\}P$, because in fact Γ and P are the same as $\{z/y\}\Gamma$ and $\{z/y\}P$, respectively.
2. However, if $y \in \mathrm{dom}(\Gamma)$, with $\Gamma = \Delta \uplus y : \alpha^{(\kappa_1,\kappa_2)}$, we can set the type of z to be $\tau = \alpha^{(0,0)}$ (since $z \notin \mathrm{dom}(\Gamma)$) and $\rho = \alpha^{(\kappa_1,\kappa_2)}$ that satisfies the conditions of lemma 42, that is, there exists ρ' and τ', namely $\rho' = \rho$ and $\tau' = \tau$, that satisfies $\rho' \preceq \rho$, $\tau' \preceq \tau$, $z : \tau$ defined (since α is defined by hypothesis) and $\tau' + \rho'$ defined. In fact, τ' and ρ' differ from each other only in

their outermost uses and $\tau' + \rho' = \rho' = \rho$. Then, by lemma 42, we have that $\Gamma = \Delta \uplus z : \alpha^{(\kappa_1,\kappa_2)} \vdash \{z/y\}P$, which is the same as $\{z/y\}\Gamma \vdash \{z/y\}P$.

This concludes the prove. □

The following definition makes precise the meaning of α-converting a bound name in a process.

Definition 45. *Let $\equiv_\alpha$ be the least congruence relation closed for the following rules:*

1. $\mathsf{new}\ x : \rho\ P \equiv_\alpha \mathsf{new}\ y : \rho\ \{y/x\}P$,
2. $a\,?\,\{l\,(\tilde{x}) = P, M\} \equiv_\alpha a\,?\,\{l\,(\tilde{y}) = \{\tilde{y}/\tilde{x}\}P, M\}$,
3. $\mathsf{def}\ X\,(\tilde{x}) = P\ \mathsf{and}\ D\ \mathsf{in}\ Q \equiv_\alpha \mathsf{def}\ X\,(\tilde{y}) = \{\tilde{y}/\tilde{x}\}P\ \mathsf{and}\ D\ \mathsf{in}\ Q$,

with $y, \tilde{y} \notin \mathrm{fn}(P)$.

The following result states that typings are preserved by α-conversion.

Lemma 46. *If $P \equiv_\alpha Q$ and $\Gamma \vdash P$, then $\Gamma \vdash Q$.*

Proof. By induction on the structure of the derivation of the typing of P. We treat all the possible cases for the final step of the type inference on P.

1. Case Msg, Nil, or App. In all these cases P is congruent to a process that has no bound names ($\mathrm{bn}(P) = \emptyset$). Hence, no renaming can take place and P and Q coincide. Then, its trivial that $\Gamma \vdash Q$.
2. Case Res. We have two subcases:
   (a) When $P \equiv \mathsf{new}\ x : \rho\ P_1 \equiv_\alpha Q \equiv \mathsf{new}\ y : \rho\ \{y/x\}P_1$, *i.e.*, the rename occurs on the restricted name.
   But $\Gamma \vdash \mathsf{new}\ x : \rho\ P_1$, then
   $$\textsc{Res}\ \frac{\Gamma \uplus x : \rho \vdash P_1}{\Gamma \vdash \mathsf{new}\ x : \rho\ P_1}.$$
   Let $y \notin \mathrm{fn}(P_1)$. Then, by lemma 44 and *R*Es,
   $$\textsc{Res}\ \frac{\text{lem 44}\ \dfrac{\Gamma \uplus x : \rho \vdash P_1}{\Gamma \uplus y : \rho \vdash \{y/x\}P_1}}{\Gamma \vdash \mathsf{new}\ y : \rho\ \{y/x\}P_1}.$$
   (b) When $P \equiv \mathsf{new}\ x : \rho\ P_1 \equiv_\alpha Q \equiv \mathsf{new}\ x : \rho\ P_1'$, with $P_1 \equiv_\alpha P_1'$. Then, by Res there exists $\Gamma \uplus x : \rho \vdash P_1$, that by induction hypothesis typifies also P_1'. Finally by Res rule
   $$\textsc{Res}\ \frac{\Gamma \uplus x : \rho \vdash P_1'}{\Gamma \vdash \mathsf{new}\ x : \rho\ P_1'}.$$

3. Case Obj. As with Res, for $P \equiv a\,?\,\{l\,(\tilde{x}) = R, M\}$ we have also two subcases to analyse.

   (a) When the renaming occurs on the abstraction names, that is, $P \equiv_\alpha Q \equiv a\,?\,\{l\,(\tilde{y}) = \ \{\tilde{y}/\tilde{x}\}R, D\}$.
   As $\Gamma \vdash P$, then

$$\textsc{Obj}\ \frac{\textsc{Meth}\ \dfrac{\Delta \uplus \tilde{x} : \tilde{\rho} \vdash R, \cdots}{\Delta \vdash \{l\,(\tilde{x}) = R, M\} : \alpha}}{\Delta + a : \alpha^{(1,0)} \vdash a\,?\,\{l\,(\tilde{x}) = R, M\}},$$

   where $\Gamma = \Delta + a : \alpha^{(1,0)}$.
   Let $y \notin \mathrm{fn}(P_1)$, then the following derivation holds

$$\textsc{Obj}\ \frac{\textsc{Meth}\ \dfrac{\text{lem 44}\ \dfrac{\Delta \uplus \tilde{x} : \tilde{\rho} \vdash R, \cdots}{\Delta \uplus \tilde{y} : \tilde{\rho} \vdash \{\tilde{y}/\tilde{x}\}R}}{\Delta \vdash \{l\,(\tilde{y}) = \{\tilde{y}/\tilde{x}\}R, M\} : \alpha}}{\Delta + a : \alpha^{(1,0)} \vdash a\,?\,\{l\,(\tilde{y}) = \{\tilde{y}/\tilde{x}\}R, M\}},$$

   that is, $\Gamma \vdash Q$.

   (b) The second case is the result of a renaming inside R.
   Then $P \equiv_\alpha a\,?\,\{l\,(\tilde{x}) = S, M\}$ and $S \equiv_\alpha R$. From induction hypothesis and Obj rule, we conclude that

$$\textsc{Obj}\ \frac{\Delta \vdash \{l\,(\tilde{x}) = S, M\} : \alpha}{\Delta + a : \alpha^{(1,0)} \vdash a\,?\,\{l\,(\tilde{x}) = S, M\}}.$$

4. Case inferred by Def. Then, P is a definition process congruent to $\mathsf{def}\ X_1\,(\tilde{x}_1) = P_1\ \mathsf{and} \ldots \mathsf{and}\ X_n\,(\tilde{x}_n) = P_n\ \mathsf{in}\ R$. This case has three subcases handled just as above for Obj and Res. Nevertheless we show their proof.

   (a) The first subcase consider the renaming of names bound by abstraction. Then, $P \equiv Q \equiv_\alpha \mathsf{def}\ X_1\,(\tilde{y}_1) = \{\tilde{y}_1/\tilde{x}_1\}P_1 \cdots$.
   Since $\Gamma \vdash P$, we have

$$\textsc{Def}\ \frac{\begin{array}{c}\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1 \\ \vdots \\ \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n \\ \Delta \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \vdash R\end{array}}{\Gamma \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1 \cdots\ \mathsf{in}\ R},$$

   with $\Gamma = \sum_i \mathcal{U}(X_i, D, Q) \times \Gamma_i + \Delta$.
   Let $y \notin \mathrm{fn}(P_1)$, then by lemma 44

$$\text{lem 44}\ \frac{\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_1 : \tilde{\rho}_1 \vdash P_1}{\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{y}_1 : \tilde{\rho}_1 \vdash \{\tilde{y}_1/\tilde{x}_1\}P_1},$$

   and by Def

$$\textsc{Def}\ \frac{\begin{array}{c}\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{y}_1 : \tilde{\rho}_1 \vdash \{\tilde{y}_1/\tilde{x}_1\}P_1 \\ \vdots \\ \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n \\ \Delta \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \vdash R\end{array}}{\Gamma \vdash \mathsf{def}\ X_1\,(\tilde{y}_1) = \{\tilde{y}_1/\tilde{x}_1\}P_1 \cdots \ \mathsf{in}\ R},$$

we conclude that $\Gamma \vdash Q$.

Notice that there is no rename of process variables and then the instantiation schema do not change, which means that $X(X_i, D, Q)$ yields the same results before and after α-conversion. This arguments applies to all the three subcases.

(b) This subcase refers to the α-congruence inside P_1. Then $P \equiv Q \equiv_\alpha \mathsf{def}\ X_1\,(\tilde{x}_1) = S_1 \cdots \ \mathsf{in}\ R$ and $S_1 \equiv_\alpha P_1$. Since $\Gamma \vdash P$, then by Def rule and induction hypothesis, we have that

$$\textsc{Def}\ \frac{\begin{array}{c}\Gamma_1 \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \cdot \tilde{x}_1 : \tilde{\rho}_1 \vdash S_1 \\ \vdots \\ \Gamma_n \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n \\ \Delta \uplus X_1 : \tilde{\rho}_1 \uplus \cdots \uplus X_n : \tilde{\rho}_n \vdash R\end{array}}{\Gamma \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = S_1 \cdots \ \mathsf{in}\ R},$$

that is, $\Gamma \vdash Q$.

(c) The last subcase consider the α-congruence inside Q. The arguments to prove this subcase are exactly the same as the ones used to prove the previous subcase and thus we omit the prove.

5. Case inferred by Par. Then, $P \equiv P_1 \,|\, P_2$. By hypothesis $\Gamma \vdash P_1 \,|\, P_2$, then there exists Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 + \Gamma_2$ and

$$\textsc{Par}\ \frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma \vdash P_1 \,|\, P_2}.$$

Q ($\equiv_\alpha P$) is of the form $Q_1 \,|\, Q_1$, where $Q_1 \equiv_\alpha P_1$ and $Q_2 \equiv_\alpha P_2$. By induction hypothesis $\Gamma_1 \vdash Q_1$ and $\Gamma_2 \vdash Q_2$, then by Par, $\Gamma \vdash Q_1 \,|\, Q_2 \equiv Q$.

Since no type derivation ends with the Meth rule, we conclude the prove. □

In the following lemma one proves that if two processes are congruent, say $P \equiv Q$, then they are typified by the same type environment, that is, if $\Gamma \vdash P$ and $P \equiv Q$, then Γ must also typify Q.

Lemma 16 (Congruence preserves typings). *If $P \equiv Q$ and $\Gamma \vdash P$, then $\Gamma \vdash Q$.*

Proof. By induction on the structure of the derivation of the typing of P. We analyse every case where two processes can be congruent (see subsection 4.1, page 11). Suppose that $\Gamma \vdash P$, then one has the following cases

1. Case $P \equiv_\alpha Q$. From lemma 46.
2. Case $P \equiv P_1 \mid P_2$. We have the fact that $\Gamma \vdash P_1 \mid P_2$, then, by the Par rule, there exists Γ_1 and Γ_2 such that

   $$\Gamma_1 \vdash P_1, \quad \Gamma_2 \vdash P_2 \quad \text{and} \quad \Gamma = \Gamma_1 + \Gamma_2.$$

   By lemma 3, we know that $+$ is commutative, which means that $\Gamma = \Gamma_1 + \Gamma_2 = \Gamma_2 + \Gamma_1$. Thus, Γ typify $P_2 \mid P_1$ as follows

   $$\textsc{Par}\ \frac{\Gamma_2 \vdash P_2 \qquad \Gamma_1 \vdash P_1}{\Gamma_2 + \Gamma_1 \vdash P_2 \mid P_1}.$$

3. Case $P \equiv (P_1 \mid P_2) \mid P_3$. If $\Gamma \vdash (P_1 \mid P_2) \mid P_3$, then there exists Γ_1, Γ_2 and Γ_3 such that $\Gamma_1 \vdash P_1$, $\Gamma_2 \vdash P_2$, $\Gamma_3 \vdash P_3$ and $\Gamma = (\Gamma_1 + \Gamma_2) + \Gamma_3$, obtained by the following type derivation

   $$\textsc{Par}\ \frac{\textsc{Par}\ \dfrac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 \mid P_2} \qquad \Gamma_3 \vdash P_3}{(\Gamma_1 + \Gamma_2) + \Gamma_3 \vdash (P_1 \mid P_2) \mid P_3}.$$

   Since $+$ is associative (lemma 3), we can write:

   $$\Gamma = (\Gamma_1 + \Gamma_2) + \Gamma_3 = \Gamma_1 + (\Gamma_2 + \Gamma_3),$$

   which typify $P \mid (Q \mid R)$. Indeed,

   $$\textsc{Par}\ \frac{\Gamma_1 \vdash P_1 \qquad \textsc{Par}\ \dfrac{\Gamma_2 \vdash P_2 \qquad \Gamma_3 \vdash P_3}{\Gamma_2 + \Gamma_3 \vdash P_2 \mid P_3}}{\Gamma_1 + (\Gamma_2 + \Gamma_3) \vdash P_1 \mid (P_2 \mid P_3)}.$$

4. Case $P \equiv P \mid \mathsf{0}$. Since any environment Δ typifies $\mathsf{0}$ by Nil rule, then particularly $\emptyset \vdash \mathsf{0}$. By hypothesis $\Gamma \vdash P$, then we can conclude that

   $$\textsc{Par}\ \frac{\Gamma \vdash P \qquad \emptyset \vdash \mathsf{0}}{\Gamma = \Gamma + \emptyset \vdash P \mid \mathsf{0}}.$$

5. Case $P \equiv \mathsf{new}\ x : \rho\ \mathsf{0}$. If $\Gamma \vdash \mathsf{new}\ x : \rho\ \mathsf{0}$, then $\Gamma \vdash \mathsf{0}$, because every environment typifies $\mathsf{0}$ (by Nil rule).
6. Case $P \equiv \mathsf{new}\ x : \rho\ \mathsf{new}\ y : \sigma\ P_1$. We have that the judgement $\Gamma \uplus x : \rho \uplus y : \sigma \vdash P_1$ hold by the following derivation

   $$\textsc{Res}\ \frac{\textsc{Res}\ \dfrac{\Gamma \uplus x : \rho \uplus y : \sigma \vdash P_1}{\Gamma \uplus x : \rho \vdash \mathsf{new}\ y : \sigma\ P_1}}{\Gamma \vdash \mathsf{new}\ x : \rho\ \mathsf{new}\ y : \sigma\ P_1}.$$

But, since the side condition imposes that $x \neq y$ or (when $x = y$) $\sigma \preceq \rho$, we also have

$$\textsc{Res}\ \frac{\textsc{Res}\ \dfrac{\Gamma \uplus x : \rho \uplus y : \sigma \vdash P_1}{\Gamma \uplus y : \sigma \vdash \mathsf{new}\ x : \rho\ P_1}}{\Gamma \vdash \mathsf{new}\ y : \sigma\ \mathsf{new}\ x : \rho\ P_1},$$

that concludes the case.

7. Case $P \equiv \Gamma \vdash \mathsf{def}\ D\ \mathsf{in}\ \mathsf{0}$. Since any type environment typifies $\mathsf{0}$, we have $\Gamma \vdash \mathsf{0}$ (by Nil).

8. Case ($P \equiv \mathsf{def}\ D\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q$). As by hypothesis $\Gamma \vdash \mathsf{def}\ D\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q$, then there exists $\Gamma = \sum_i \mathcal{U}(X_i, D, \mathsf{new}\ x : \rho\ Q) \cdot \Gamma_i + \Delta$ such that:

$$\textsc{Def}\ \frac{\Gamma_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \textsc{Res}\ \dfrac{\Delta \uplus x : \rho \vdash Q}{\Delta \vdash \mathsf{new}\ x : \rho\ Q}}{\Gamma \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1 \cdots\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q}.$$

But, we can also have the derivation

$$\textsc{Res}\ \frac{\textsc{Def}\ \dfrac{\Gamma_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \Delta \uplus x : \rho \vdash Q}{\sum_i \mathcal{U}(X_i, D, Q) \cdot \Gamma_i + \Delta \uplus x : \rho \vdash \mathsf{def}\ X_1\,(\tilde{x}_1) = P_1 \cdots\ \mathsf{in}\ Q}}{\Gamma \vdash \mathsf{new}\ x : \rho\ \mathsf{def}\ X_1 = (\tilde{x}_1)\ P_1 \cdots\ \mathsf{in}\ Q}$$

Thus, Γ typifies the requested congruence. However, we have to force that x must not belong to any of the Γ_i, that is, $x \notin \mathrm{fn}(D)$ when applying the Def rule.

9. Case $P \equiv (\mathsf{def}\ D\ \mathsf{in}\ Q) \mid R$. By hypothesis $\Gamma \vdash (\mathsf{def}\ D\ \mathsf{in}\ Q) \mid R$. Then,

$$\textsc{Par}\ \frac{\textsc{Def}\ \dfrac{\Delta_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \quad \Delta \uplus X_i : \tilde{\rho}_i \vdash Q}{\Gamma_1 \vdash \mathsf{def}\ D\ \mathsf{in}\ Q} \qquad \Gamma_2 \vdash R}{\Gamma \vdash (\mathsf{def}\ D\ \mathsf{in}\ Q) \mid R},$$

where $\Gamma = \Gamma_1 + \Gamma_2$ and $\Gamma_1 = \sum_i \mathcal{U}(X_i, D, Q) \cdot \Delta_i + \Delta$.
But,

$$\textsc{Def}\ \frac{\Delta_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \textsc{Par}\ \dfrac{\Delta \uplus X_i : \tilde{\rho}_i \vdash Q \quad \Gamma_2 \vdash R}{\Delta \uplus X_i : \tilde{\rho}_i + \Gamma_2 \vdash Q \mid R}}{\Gamma = \sum_i \mathcal{U}(X_i, D, Q \mid R) \cdot \Delta_i + \Delta + \Gamma_2 \vdash \mathsf{def}\ D\ \mathsf{in}\ (Q \mid R)}.$$

However, for the typing to be correct, we must enforce that no $\mathrm{fv}(R)$ gets bound by def. Notice that this condition also enforces that $\mathcal{U}(X_i, D, Q) = \mathcal{U}(X_i, D, Q \mid R)$ for all $1 \leq i \leq n$, since no process variable in R is equal to a particular X_i.

This concludes the proof. □

Proof of theorem 19 (subject reduction), page 14. By induction on the structure of the derivation of the typing of P. We analyse the last typing rule applied.

1. Case Com. By hypothesis, one knows that

$$\Gamma \vdash a\,!\,l_i[\tilde{v}] \,|\, a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\},$$

with $\Gamma(a) = \alpha^{(\kappa_1,\kappa_2)} = \{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(\kappa_1,\kappa_2)}$. Then, the following inference holds

$$\textsc{Par}\ \frac{\textsc{Msg}\ \dfrac{}{\Gamma_1 \vdash a\,!\,l_i[\tilde{v}]}\quad \textsc{Obj}\ \dfrac{\textsc{Meth}\ \dfrac{\Delta \uplus \tilde{x}_1 : \tilde{\tau}_1 \vdash P_1 \quad \cdots \quad \Delta \uplus \tilde{x}_1 : \tilde{\tau}_n \vdash P_n}{\Delta \vdash \{l_1\,(\tilde{x}_i) = P_1, \cdots\} : \{l_1 : \tilde{\tau}_1, \cdots\}^{(1,0)}}}{\Gamma_2 \vdash a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}}}{\Gamma \vdash a\,!\,l_i[\tilde{v}] \,|\, a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}},$$

with

$$\begin{array}{l} \Gamma \preceq \Gamma_1 + \Gamma_2, \\ \Gamma_1 \preceq a : \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i, \\ \Gamma_2 \preceq \Delta + a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(1,0)}, \\ \Gamma(a) = \alpha^{(\kappa_1,\kappa_2)} \preceq \Gamma_1(a) = \alpha^{(\kappa_{11},\kappa_{12})} + \Gamma_2(a) = \alpha^{(\kappa_{21},\kappa_{22})}. \end{array} \tag{2}$$

First, we show that κ_1^- and κ_2^- are defined. From $\Gamma_1(a) = \alpha^{(\kappa_{11},\kappa_{12})} \preceq a : \{l_1 : \tilde{\rho}_1, \cdots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i$, we get that $\kappa_{12} \geq 1$. Using a similar argument, we can show that $\kappa_{21} \geq 1$, because $\Gamma_2(a) = \alpha^{(\kappa_{21},\kappa_{22})} \preceq \Delta + a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(1,0)}$. Hence, $\kappa_1 \geq 1$ and $\kappa_2 \geq 1$, and thus, $\Gamma(a) = \alpha^{(\kappa_1^-,\kappa_2^-)}$ is defined.
Next, we show that $\Gamma^{-\ell} \vdash \{\tilde{v}/\tilde{x}_i\}P_i$. Possible values for (κ_1, κ_2), are $(1,1), (1,\omega), (\omega,1)$, and (ω,ω). We analyse the "worse" case (when $\kappa_1 = \kappa_2 = 1$). Let $\Gamma(a) = \alpha^{(1,1)}$. From the type inference of $\Gamma \vdash P$, we conclude that $\Delta(a) \preceq \tau^{(0,0)}$. Since $\Gamma \preceq \Gamma_1 + \Gamma_2$, we find that (i) $\tilde{\rho}_i \preceq \tilde{\sigma}_i$, because $\{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(1,1)} \preceq \{l_1 : \tilde{\rho}_1, \cdots, \tilde{\rho}_n\}^{(0,1)}$ (subtyping is contra-variant on output), and (ii) $\tilde{\sigma}_i \preceq \tilde{\tau}_i$, because $\{l_1 : \tilde{\sigma}_1, \cdots, l_n : \tilde{\sigma}_n\}^{(1,1)} \preceq \{l_1 : \tilde{\tau}_1, \cdots, \tilde{\tau}_n\}^{(1,0)}$ (subtyping is covariant on input). Thus, $\tilde{\rho}_i \preceq \tilde{\sigma}_i \preceq \tilde{\tau}_i$ and hence $\tilde{\rho}_i \preceq \tilde{\tau}_i$.
From (2), we know that $\Gamma_1 \backslash a \preceq \tilde{v} : \tilde{\rho}_i$ and that $(\Gamma_1 \backslash a) + \Delta \uplus \tilde{x}_i : \tilde{\rho}_i$ is defined. Then, by lemma 43,

$$(\Gamma_1 + \Gamma_2) \setminus a \uplus a : \{l_1 : \tilde{\tau}_1, \cdots, l_n : \tilde{\tau}_n\}^{(0,0)} \vdash \{\tilde{v}/\tilde{x}_i\}P_i.$$

The analysis of the remaining cases is similar.

2. Case by Par. If $\Gamma \vdash P \,|\, Q$, then, by lemma 44, there exists Γ_1 and Γ_2 such that $\Gamma \preceq \Gamma_1 + \Gamma_2$, with $\Gamma_1 \vdash P$ and $\Gamma_2 \vdash Q$. We have to consider two subcases: for $\ell = x$ and for $\ell = \epsilon$.

(a) Suppose $\ell = x$. Thus, $\Gamma \uplus x : \alpha^{(\kappa_1,\kappa_2)} \vdash P \,|\, Q$ and $\Gamma_1 \uplus x^{(\kappa_3,\kappa_4)} \vdash P$. But, by hypothesis, one knows that $\Gamma_1 \uplus x^{(\kappa_3^-,\kappa_4^-)} \vdash R$, and then, by PAR rule, $\Gamma_1 \uplus x^{(\kappa_3^-,\kappa_4^-)} + \Gamma_2 \vdash R \,|\, Q$. We conclude that $\Gamma \uplus x^{(\kappa_1^-,\kappa_2^-)}$ is defined, and that $\Gamma \uplus x^{(\kappa_1^-,\kappa_2^-)} \preceq \Gamma_1 \uplus x^{(\kappa_3^-,\kappa_4^-)} + \Gamma_2$. Hence,

$$\Gamma \uplus x^{(\kappa_1^-,\kappa_2^-)} \vdash R \,|\, Q.$$

(b) When $\ell = \epsilon$, we have that $\Gamma_1 \uplus x^{(\kappa_3,\kappa_4)} \vdash P$, and by hypothesis also typify R (because its an ϵ transition), then $\Gamma \uplus x^{(\kappa_1,\kappa_2)} \vdash R \,|\, Q$.

3. Case RES$_1$. Then we have $P \equiv \mathsf{new}\ x : \alpha^{(\kappa_1,\kappa_2)}\ P_1$. Since $\Gamma \vdash P$, then, by RES rule, $\Gamma \uplus x : \alpha^{(\kappa_1,\kappa_2)}\ P_1$ and by induction hypothesis, $\Gamma \uplus x : \alpha^{(\kappa_1^-,\kappa_2^-)} \vdash R$. Applying again RES rule, we get

$$\Gamma \vdash \mathsf{new}\ x : \alpha^{(\kappa_1^-,\kappa_2^-)}\ R.$$

Notice that the use of resources is registered on the type of x (and not on the environment), because the transition is by ϵ.

4. Case RES$_2$. By RES rule, we know, since $\Gamma \vdash \mathsf{new}\ x : \rho\ P_1$, that $\Gamma \uplus x : \rho \vdash P_1$. By induction hypothesis, $\Gamma^{-\ell} \uplus x : \rho \vdash R$. Thus by RES rule, we obtain

$$\Gamma^{-\ell} \vdash \mathsf{new}\ x : \rho\ R, \quad \text{with } \Gamma^{-\ell} \text{ defined by IH.}$$

5. Case STR. From hypothesis $\Gamma \vdash P$ and by lemma 16, we conclude that $\Gamma \vdash R$. By induction hypothesis $\Gamma^{-\ell} \vdash S$ and again by lemma 16, $\Gamma^{-\ell} \vdash Q$. Hence, $\Gamma^{-\ell}$ is defined and typifies Q.

6. Case inferred by DEF. We have, from hypothesis, that $\Gamma \vdash \mathsf{def}\ D\ \mathsf{in}\ Q$ and $Q \xrightarrow{\ell} R$. By DEF rule, there exists Γ_i and Δ, such that $\Gamma_i \uplus \tilde{X} : \tilde{\rho} \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i$ and $\Delta \uplus \tilde{X} : \tilde{\rho} \vdash Q$, with $\Gamma = \sum_i \mathcal{U}(X_i, D, Q) \cdot \Gamma_i + \Delta$. Induction hypothesis guaranties that $\Delta^{-\ell}$ is defined and $\Delta^{-\ell} \uplus \tilde{X} : \tilde{\rho} \vdash R$. Thus, the following derivation holds

$$\textsc{Def}\ \frac{\Gamma_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \Delta^{-\ell} \uplus \tilde{X} : \tilde{\rho} \vdash R}{\Pi \vdash \mathsf{def}\ D\ \mathsf{in}\ R},$$

where $\Pi = \mathcal{U}(X_i, D, R) \cdot \Gamma_i + \Delta^{-\ell}$. However, it remains to show that $\Gamma^{-\ell} \vdash \mathsf{def}\ D\ \mathsf{in}\ R$. First of all, we know that U (see definition 9) satisfies the condition $\mathcal{U}(X_i, D, Q) \geq \mathcal{U}(X_i, D, R)$, when $Q \xrightarrow{\ell} R$. On the other hand, as the reduction occurs in R, $\Pi(\ell) = \Gamma^{-\ell}(\ell)$, since Π results from the summation $\Delta^{-\ell}$. Thus, $\Pi \vdash P$ and $\Gamma^{-\ell} \preceq \Pi$, then by successive applications of WEAK and RES rules (to Π), $\Gamma^{-\ell} \vdash P$, which concludes the case.

7. Case CALL. If $\Gamma \vdash \mathsf{def}\ D\ \mathsf{in}\ X_i[\tilde{v}] \,|\, Q$, then

$$\textsc{Def}\ \frac{\Gamma_i \uplus \tilde{X} : \tilde{\rho} \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \textsc{Par}\ \dfrac{\textsc{App}\ \dfrac{}{X_i : \tilde{\rho}_i \uplus \tilde{v} : \tilde{\rho}_i \vdash X_i[\tilde{v}]} \quad p\Delta \uplus \tilde{X} : \tilde{\rho} \vdash Q}{\Delta \uplus \tilde{X} : \tilde{\rho} + \tilde{v} : \tilde{\tau} \vdash X_i[\tilde{v}] \,|\, Q}}{\Gamma \vdash \mathsf{def}\ D\ \mathsf{in}\ X_i[\tilde{v}] \,|\, Q},$$

where, by lemma 40, $\Gamma \preceq \mathcal{U}(X_i, D, X_i[\tilde{v}] \,|\, Q) \cdot \Gamma_i + \Delta + \tilde{v} : \tilde{\tau}$ and $\Delta \uplus X_i : \tilde{\rho} + \tilde{v} : \tilde{\tau} \preceq \Delta \uplus X_i : \tilde{\rho}_i + \tilde{v} : \tilde{\rho}$. Hence, $\tau \preceq \rho$ and $v_1 : \tau_1 + \cdots + v_n : \tau_n$ are defined. Then, by lemma 43,

$$\Gamma_i \uplus \tilde{X} : \tilde{\rho} + \tilde{v} : \tilde{\tau} \vdash \{\tilde{v}/\tilde{x}\}P_i$$

and hence,

$$\textsc{Def}\ \frac{\Gamma_i \uplus X_i : \tilde{\rho}_i \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i \qquad \textsc{Par}\ \dfrac{\Gamma_i \uplus \tilde{X} : \tilde{\rho} + \tilde{v} : \tilde{\tau} \vdash \{\tilde{v}/\tilde{x}_i\}P_i \quad \Delta \uplus \tilde{X} : \tilde{\rho} \vdash Q}{\Lambda \uplus \tilde{X} : \tilde{\rho} + \tilde{v} : \tilde{\tau} \vdash \{\tilde{v}/\tilde{x}_i\}P_i \,|\, Q}}{\Pi \vdash \mathsf{def}\ D\ \mathsf{in}\ \{\tilde{v}/\tilde{x}_i\}P_i \,|\, Q}.$$

It remains to show that Γ and Π denote the same type environment. Let R and S denote, respectively, the process expressions $X_i[\tilde{v}] \,|\, Q$ and $\{\tilde{v}/\tilde{x}_i\}P_i \,|\, Q$. The definitions of Γ and Π are

$$\begin{aligned}\Gamma &= \mathcal{U}(X_1, D, R) \cdot \Gamma_1 + \cdots + \mathcal{U}(X_n, D, R) \cdot \Gamma_n + \Delta + \tilde{v} : \tilde{\tau} \\ \Pi &= \mathcal{U}(X_1, D, S) \cdot \Gamma_1 + \cdots + \mathcal{U}(X_n, D, S) \cdot \Gamma_n + \Gamma_i + \Delta + \tilde{v} : \tilde{\tau}\end{aligned}$$

Recall that function U satisfies definition 9. Hence, for $X_j \neq X_i$, $U_{X_j}(\mathsf{def}\ D\ \mathsf{in}\ R) = U_{X_j}(\mathsf{def}\ D\ \mathsf{in}\ S)$, and $U_{X_i}(\mathsf{def}\ D\ \mathsf{in}\ R) = 1 + U_{X_i}(\mathsf{def}\ D\ \mathsf{in}\ S)$. But Π as an explicitly summation of Γ_i, then $\Gamma = \Pi$.

This concludes the proof. □

A.2 Proofs for section 5

In this section we prove to main results: the equivalence of the two types systems and the correction of the LTR algorithm.

Proof of theorem 27 (equivalence of the two type systems), page 17. Assertion one and two are proved by induction on the structure of the derivation of the typing of P. We analyse the last typing rule applied.

Proof of assertion 1.

1. Case $\textsc{Msg}_{\textsc{sd}}$. Let S be a solution of C, $\Gamma(a) = \tau = \{l_i : \tilde{\tau}_i\}^{(u_1,u_2)}_{1 \leq i \leq n}$, and $\Gamma(\tilde{v}) = \tilde{\tau}'$. Then, $S\tau \preceq S\rho$ for $\rho = \{l_i : \tilde{\rho}_i\}^{(0,1)}_{1 \leq i \leq n}$ and

$S\tilde{\tau}' \preceq S\tilde{\rho}_i$ holds. Applying Msg rule once and Sub rule twice we obtain

$$\dfrac{\dfrac{\dfrac{}{a : S\rho + \tilde{v} : S\tilde{\rho}_i \vdash a\,!\,l_i[\tilde{v}]} \qquad S\tau \preceq S\rho}{a : S\tau + \tilde{v} : S\tilde{\rho}_i \vdash a\,!\,l_i[\tilde{v}]} \qquad \tilde{\tau}' \preceq \tilde{\rho}_i}{a : S\{l_i : \tilde{\tau}_i\}_{1\leq i\leq n}^{(u_1,u_2)} + \tilde{v} : S\tilde{\tau}' \vdash a\,!\,l_i[\tilde{v}]}$$

Successive applications of Weak conclude the case.

2. Case Par_{SD}. Let S be a solution of C. Since $C \models C_1 \cup C_2$, then, by proposition 26, S is a solution of C_1 and also a solution of C_2. Applying induction hypothesis $S\Gamma_1 \vdash SP_1$ and $S\Gamma_2 \vdash SP_2$. Hence, by Par rule, $S(\Gamma_1 + \Gamma_2) \vdash S(P_1 \mid P_2)$. The result $S\Gamma \vdash S(P_1 \mid P_2)$ follows by successive applications of Weak rule.

3. Case Res_{SD}. By induction hypothesis $S(\Gamma \uplus x : \sigma) \vdash SP$ for S a solution of C. Since $S\rho \preceq S\sigma$, then

$$\text{New}\ \dfrac{\text{Sub}\ \dfrac{S(\Gamma \uplus x : \sigma) \vdash SP \qquad S\rho \preceq S\sigma}{S(\Gamma \uplus x : \rho) \vdash P}}{S\Gamma \vdash S(\mathsf{new}\ x : \rho\ P)}$$

4. Case Obj_{SD}. Let S be a solution of C and $\alpha = \{l_i : \tilde{\rho}_i\}_{1\leq i\leq n}$. Then S is a solution of C', because $C \models C'$, and by induction hypothesis $S\Delta \vdash SM : S\alpha$. From $C \models \Gamma \preceq \Delta + a : \alpha^{(1,0)}$, $S\Gamma(a) = \{l_i : \tilde{\tau}_i\}_{1\leq i\leq n}^{(\kappa_1,\kappa_2)} \preceq S(\Delta(a) + \alpha^{(1,0)})$. The subtype relation is covariant on input, then $\tilde{\tau}_i \preceq \tilde{\rho}_i$ for $(1 \leq i \leq n)$. Hence, the following derivation holds.

$$\text{Obj}\ \dfrac{\text{Sub}\ \dfrac{S\Delta \vdash SM : S\alpha \qquad \tilde{\tau}_i \preceq \tilde{\rho}_i}{S\Delta \vdash SM : S(\{l_i : \tilde{\tau}_i\}_{1\leq i\leq n})}}{S\Delta + a : S(\{l_i : \tilde{\tau}_i\}_{1\leq i\leq n}^{(1,0)}) \vdash a\,?\,\{M\}}$$

Applying Weak we conclude the case.

5. Case App_{SD}. Let S be a solution of C. By App and Sub rules

$$\text{Sub}\ \dfrac{\text{App}\ \dfrac{}{X : S\tilde{\rho} \uplus \tilde{v} : S\tilde{\rho} \vdash X[\tilde{v}]} \qquad S\tilde{\sigma} \preceq S\tilde{\rho}}{X : S\tilde{\rho} \uplus \tilde{v} : S\tilde{\sigma} \vdash X[\tilde{v}]}$$

The case concludes by successive applications of Weak.

6. Case Nil_{SD}. Since $\emptyset \vdash \mathsf{0}$, also $S\Gamma \vdash \mathsf{0}$ by successive applications of Weak rule.

7. Case Meth_{SD}. Let S be a solution of C. Since $C \models \bigcup_j C_j$ for $1 \leq j \leq n$, then, by induction hypothesis, $S(\Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i) \vdash SP_i$

for $1 \leq i \leq n$. But $S\tilde{\rho}_i \preceq S\tilde{\sigma}_i$ $(1 \leq i \leq n)$, then applying Sub and Meth rules

$$\textsc{Meth}\ \frac{\textsc{Sub}\ \dfrac{S(\Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i) \vdash SP_i \qquad S\tilde{\rho}_i \preceq S\tilde{\sigma}_i, 1 \leq i \leq n}{S(\Gamma_i \uplus \tilde{x}_i : \tilde{\rho}_i) \vdash SP_i, \quad 1 \leq i \leq n}}{S\Gamma \vdash S(l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n) : S(\{l_1 : \tilde{\rho}_i\}_{1 \leq i \leq n})}$$

By successive applications of Sub and Weak we conclude the case, since $C \models \bigcup_j (\Gamma \preceq \Gamma_j)$ for $1 \leq j \leq n$ (and S is a solution of C).

that concludes the proof of assertion 1.

Next the proof of assertion 2.

1. Case Par. Since $\Gamma \vdash P$, by hypothesis, there exists Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 + \Gamma_2$, $\Gamma_1 \vdash P_1$, and $\Gamma_2 \vdash P_2$. By induction hypothesis $\Gamma_1; \emptyset \vdash P_1$ and $\Gamma_2; \emptyset \vdash P_2$, then $\emptyset \models C_1 \cup C_2 = \emptyset$ and $\emptyset \models \Gamma \preceq \Gamma_1 + \Gamma_2$ since, by hypothesis, $\Gamma = \Gamma_1 + \Gamma_2$ and $\preceq$ is reflexive. Hence,

$$\textsc{Par}_{\textsc{sd}}\ \frac{\Gamma_1; \emptyset \vdash P \quad \Gamma_2; \emptyset \vdash Q \quad C \models \emptyset \cup (\Gamma \preceq \Gamma_1 + \Gamma_2)}{\Gamma; \emptyset \vdash P \mid Q}$$

2. Case Msg. Then $\Gamma \vdash a\,!\,l_i[\tilde{v}]$ and the following derivation holds.

$$\textsc{Sub}\ \frac{\textsc{Msg}\ \dfrac{}{a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i \vdash a\,!\,l_i[\tilde{v}]} \quad \tilde{\sigma} \preceq \tilde{\rho}_i}{a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\sigma} \vdash a\,!\,l_i[\tilde{v}]}$$

that by successive applications of Weak and Sub rules we achieve that $\Gamma \vdash a[\tilde{v}]$ where $\Gamma(a) = \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(\kappa_1, \kappa_2)}$ (with $\kappa_2 \geq 1$) and $\Gamma(\tilde{v}) = \tilde{\sigma}$ (with $\tilde{\sigma} \preceq \tilde{\rho}_i$). Thus, $\Gamma \preceq a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i$ holds by hypothesis. Hence, $\Gamma; \emptyset \vdash a\,!\,l_i[\tilde{v}]$.

3. Case New. By hypothesis the following derivation holds.

$$\textsc{New}\ \frac{\textsc{Sub}\ \dfrac{\Gamma \uplus x : \sigma \vdash P_1 \qquad \rho \preceq \sigma}{\Gamma \uplus x : \rho \vdash P_1}}{\Gamma \vdash \mathsf{new}\ x : \rho\ P_1}$$

Then, by induction hypothesis, $\Gamma \uplus x : \sigma; \emptyset \vdash P_1$. Since $\rho \preceq \sigma$ holds by hypothesis, then $\emptyset \models \rho \preceq \sigma$. Therefore,

$$\textsc{Res}_{\textsc{sd}}\ \frac{\Gamma \uplus x : \sigma; \emptyset \vdash P_1 \qquad \emptyset \models \rho \preceq \sigma}{\Gamma; \emptyset \vdash \mathsf{new}\ x : \rho\ P_1}$$

4. Case Nil. By hypothesis $\Gamma \vdash \mathsf{0}$, then by $\textsc{Res}_{\textsc{sd}}$ rule $\Gamma; \emptyset \vdash \mathsf{0}$.

5. Case App. Since $\Gamma \vdash X[\tilde{v}]$, then there exists a type derivation such that

$$\textsc{Sub}\ \frac{X : \tilde{\rho} \uplus \tilde{v} : \tilde{\rho} \vdash X[\tilde{v}] \qquad \tilde{\sigma} \preceq \tilde{\rho}}{X : \tilde{\rho} \uplus \tilde{v} : \tilde{\sigma} \vdash X[\tilde{v}]}$$

and by successive applications of Weak rule, $\Gamma \vdash X[\tilde{v}]$. As $\tilde{\sigma} \preceq \tilde{\rho}$ holds by hypothesis, then $\emptyset \models \tilde{\sigma} \preceq \tilde{\rho}$. Therefore, $\Gamma \uplus X : \tilde{\rho} \uplus \tilde{v} : \tilde{\sigma}; \emptyset \vdash X[\tilde{v}]$.

6. Case Meth. By hypothesis $\Gamma \vdash P : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}$ for $P \equiv \{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}$. Then the following derivations hold.

$$\textsc{Sub}\ \frac{\textsc{Weak}\ \dfrac{\textsc{Weak}\ \dfrac{\Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i \vdash P_i}{\vdots}}{\Gamma \uplus \tilde{x}_i : \tilde{\sigma}_i \vdash P_i} \qquad \tilde{\rho}_i \preceq \tilde{\sigma}_i}{\Gamma \uplus \tilde{x}_i : \tilde{\rho}_i \vdash P_i}$$

for $1 \leq i \leq n$. So, also holds

$$\textsc{Meth}\ \frac{\Gamma \uplus \tilde{x}_1 : \tilde{\rho}_1 \quad \ldots \quad \Gamma \uplus \tilde{x}_n : \tilde{\rho}_n \vdash P_n}{\Gamma \vdash \{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\} : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}}$$

By induction hypothesis, $\Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i; \emptyset \vdash P_i$ for $1 \leq i \leq n$. As $\tilde{\rho}_i \preceq \tilde{\sigma}_i$ for $1 \leq i \leq n$ holds by hypothesis (is a premise in the above type derivation), then $\emptyset \models \bigcup_i (\tilde{\rho}_i \preceq \tilde{\sigma}_i)$. On the other hand, Γ was constructed by successive applications of Weak and Sub rules, thus $\bigcup_i (\Gamma(x) \preceq \Gamma_i(x)$ for $x \in \mathrm{dom}(\Gamma_i)$ and $1 \leq i \leq n$. Therefore, the following derivation holds.

$$\textsc{Meth}_{\textsc{sd}}\ \frac{\Gamma_i \uplus \tilde{x}_i : \tilde{\sigma}_i; \emptyset \vdash P_i, \quad 1 \leq i \leq n \qquad \emptyset \models \bigcup_j (\{\Gamma \preceq \Gamma_j\} \cup \emptyset \cup \{\tilde{\rho}_j \preceq \tilde{\sigma}_j\})}{\Gamma; \emptyset \vdash P : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}}$$

7. Case Obj, Def. These cases are proved following similar arguments as for the previous case.

8. Case Sub. Then the following derivation holds.

$$\textsc{Sub}\ \frac{\Gamma \uplus x : \sigma \vdash P \qquad \rho \preceq \sigma}{\Gamma \uplus x : \rho \vdash P}$$

By induction hypothesis, $\Gamma \uplus x : \sigma; \emptyset \vdash P$ and as $\rho \preceq \sigma$ holds by hypothesis, then $\Gamma \uplus x : \rho; \emptyset \vdash P$.

9. Case Weak. This case is similar to previous one.

that concludes the proof. □

Proof of theorem 34 (correctness of the algorithm), page 20. For assertion 1 we proceed by induction on the structure of the derivation of the typing of P. We treat all the possible cases for the final step of the type inference on P. We show only a few cases because the other cases are similar.

1. Case $\textsc{Msg}_{\text{SD}}$. Then $\Gamma; C \vdash P$ and the following derivation holds.

$$\textsc{Msg}_{\text{SD}} \ \frac{C \models \Gamma \preceq a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i}{\Gamma; C \vdash a\,!\,l_i[\tilde{v}]}$$

   Let $\Gamma'; C' = \text{LTR}(a\,!\,l_i[\tilde{v}])$. Then, $\Gamma'; C' = a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} \oplus \tilde{v} : \tilde{\rho}_i$ that, by proposition 33, satisfies $C' \models \Gamma' \preceq a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} \oplus \tilde{v} : \tilde{\rho}_i$ and is a principal typing of P. Hence, $\Gamma; C$ is an instance of $\Gamma'; C'$.

2. Case $\textsc{Res}_{\text{SD}}$. Then $P \equiv \mathsf{new}\ x : \rho\ P_1$ and $\Gamma; C \vdash P$. We show that $\Gamma'; C' \vdash \mathsf{new}\ x : \rho\ P_1$ where $\Gamma'; C' = \text{LTR}(\mathsf{new}\ x : \rho\ P_1)$. Let $\Delta; C'' = \text{LTR}(P_1)$. Then, by induction hypothesis, $\Delta; C''$ is a principal typing of P_1.

   We consider two subcases whether x belongs or not to $\text{dom}(\Delta)$. If $x \in \text{dom}(\Delta)$, then $C'' \cup \{\rho \preceq \Delta(x)\} \models \rho \preceq \Delta(x)$ and by $\textsc{Res}_{\text{SD}}$ rule we have $\Gamma'; C' \vdash \mathsf{new}\ x : \rho\ P_1$ for $\Gamma' = \Delta \setminus x$ and $C' = C'' \cup \{\rho \preceq \Delta(x)\}$. The other subcase, where $x \notin \text{dom}(\Gamma)$, is similar.

For assertion 2 we proceed by induction on structure of P. We show only a few cases because the other cases are similar.

1. Case $P \equiv a\,!\,l_i[\tilde{v}]$. Then $\text{LTR}(a\,!\,l_i[\tilde{v}]) = \Gamma; C = a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} \oplus \tilde{v} : \tilde{\rho}_i$. By proposition 33, $C \models \Gamma \preceq a : \{l_1 : \tilde{\rho}_1, \ldots, l_n : \tilde{\rho}_n\}^{(0,1)} + \tilde{v} : \tilde{\rho}_i$ and by $\textsc{Msg}$ rule $\Gamma; C \vdash P$. Proposition 33 also guaranties that $\Gamma; C$ is a principal type of P.

2. Case $P \equiv P_1 \,|\, P_2$. We show that $\Gamma; C \vdash P_1 \,|\, P_2$ where $\Gamma; C' = \Gamma_1 \oplus \Gamma_2$, $C = C' \cup C_1 \cup C_2$, $\Gamma_1; C_1 = \text{LTR}(P_1)$, and $\Gamma_2; C_2 = \text{LTR}(P_2)$. By induction hypothesis, $\Gamma_1; C_1 \vdash P_1$, $\Gamma_2; C_2 \vdash P_2$ and $C' \models \Gamma \preceq \Gamma_1 + \Gamma_2$. Since $C = C' \cup C_1 \cup C_2$, then, by proposition 26, $C \models C_1$ and $C \models C_2$. Therefore by $\textsc{Par}_{\text{SD}}$ rule $\Gamma; C \vdash P_1 \,|\, P_2$.

The remaining cases are shown as the cases above. □

A.3 Proofs for section 6

Essentially the proof that $\mathcal{U}$ is a call-counting function.

The next result is needed in the proof of lemma 48.

Proposition 47. *Suppose that X is process variable, P a process, and $D = X_1(\tilde{x}_1) = P_1$ and $\ldots$ and $X_n(\tilde{x}_n) = P_n$ and $D' = X'_1(\tilde{x}_1) = P'_1$ and $\ldots$ and $X'_n(\tilde{x}_n) = P'_n$ two definitions such that $\mathrm{fv}(P) \cap FV(D') = \emptyset$. Then*

$$\mathcal{U}(X, D, P) = \mathcal{U}(X, D', P) + \sum_j (\mathcal{U}(X_j, D', P) \times \mathcal{U}(X, D, P_j)$$

Proof. The proof is by straightforward induction on the structure of P. The interesting cases are for $P \equiv Y[\tilde{v}]$ and $P \equiv \mathsf{def}\ D''\ \mathsf{in}\ P'$. The former is proved by an exhaustive analysis to whether or not $X = Y$ and Y is bound (or not) to D. The later is proved by using three times the induction hypothesis. □

The following lemma states that congruence preserves the number of calls to process variables.

Lemma 48. *Given a process variable X, a definition D, and process expressions P and Q, if $P \equiv Q$, then $\mathcal{U}(X, D, P) = \mathcal{U}(X, D, Q)$.*

Proof. We analyse every case where two processes can be congruent (see subsection 4.1, page 11).

1. Case $P \equiv_\alpha Q$. Since α-congruence does not affect process variables $\mathcal{U}(X, D, P) = \mathcal{U}(X, D, Q)$.
2. Case $P \mid Q \equiv Q \mid P$. Then,

$$\begin{aligned}\mathcal{U}(X, D, P \mid Q) &= \mathcal{U}(X, D, P) + \mathcal{U}(X, D, Q) \\ &= \mathcal{U}(X, D, Q) + \mathcal{U}(X, D, P) = \mathcal{U}(X, D, Q \mid P)\end{aligned}$$

   since sum is commutative (proposition 3, page 4).
3. Case $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, $P \mid \mathsf{0} \equiv P$, $\mathsf{new}\ x : \rho\ \mathsf{0} \equiv \mathsf{0}$, $\mathsf{new}\ x : \rho\ \mathsf{new}\ y : \sigma\ P \equiv \mathsf{new}\ y : \sigma\ \mathsf{new}\ x : \rho\ P$, and $(\mathsf{new}\ x : \rho\ P) \mid Q \equiv \mathsf{new}\ x : \rho\ (P \mid Q)$. These cases are handled as the previous one, considering that sum is also associative, that $\mathcal{U}(X, D, \mathsf{0}) = 0$, and that for a given process P, $\mathcal{U}(X, D, \mathsf{new}\ x : \rho\ P) = \mathcal{U}(X, D, P)$.
4. Case $\mathsf{def}\ D'\ \mathsf{in}\ \mathsf{0} \equiv \mathsf{0}$. As

$$\begin{aligned}\mathcal{U}(X, D, \mathsf{def}\ D'\ \mathsf{in}\ \mathsf{0}) &= \mathcal{U}(X, D', \mathsf{0}) \\ &\quad + \sum_j \{\mathcal{U}(X_j, D', \mathsf{0}) \times \mathcal{U}(X, D, P_j)\} \\ &= 0 = \mathcal{U}(X, D, \mathsf{0})\end{aligned}$$

5. Case $\mathsf{def}\ D\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q \equiv \mathsf{new}\ x : \rho\ \mathsf{def}\ D\ \mathsf{in}\ Q$. In fact,

$$\begin{aligned}\mathcal{U}(X, D, \mathsf{def}\ D'\ \mathsf{in}\ \mathsf{new}\ x : \rho\ Q) &= \mathcal{U}(X, D', \mathsf{new}\ x : \rho\ Q) \\ &\quad + \sum_j \{\mathcal{U}(X_j, D', \mathsf{new}\ x : \rho\ Q) \times \mathcal{U}(X, D, P_j)\} \\ &= \mathcal{U}(X, D, Q) + \sum_j \{\mathcal{U}(X_j, D', Q) \times \mathcal{U}(X, D, P_j)\} \\ &= \mathcal{U}(X, D, \mathsf{def}\ D\ \mathsf{in}\ Q) = \mathcal{U}(X, D, \mathsf{new}\ x : \rho\ \mathsf{def}\ D\ \mathsf{in}\ Q)\end{aligned}$$

6. Case $(\mathsf{def}\ D'\ \mathsf{in}\ Q) \mid R \equiv \mathsf{def}\ D'\ \mathsf{in}\ (Q \mid R)$. First notice that the $\mathrm{fv}(R)$ are not bound by D' (insured by the side condition). Therefore,

$$\begin{aligned}\mathcal{U}(X, D, (\mathsf{def}\ D'\ \mathsf{in}\ Q) \mid R) &= \mathcal{U}(X, D', Q) + \mathcal{U}(X, D, R) \\ &\quad + \sum_j \{\mathcal{U}(X_j, D', Q) \times \mathcal{U}(X, D, P_j)\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{U}(X, D, \mathsf{def}\ D'\ \mathsf{in}\ (Q \mid R)) &= \mathcal{U}(X, D', Q) + \mathcal{U}(X, D', R) \\ &\quad + \sum_j \{(\mathcal{U}(X_j, D', Q) + \mathcal{U}(X_j, D', R)) \times \mathcal{U}(X, D, P_j)\}\end{aligned}$$

But, by proposition 47, $\mathcal{U}(X, D, R) = \mathcal{U}(X, D', R) + \sum_j \{\mathcal{U}(X_j, D', Q) + \times \mathcal{U}(X, D, P_j)\}$, which concludes the proof.

□

Proof of theorem 38 ($\mathcal{U}$ is a call-counting function), page 23.

Proof of assertion 1. We proceed by induction on the structure of the reduction. We analyse the last typing rule of the derivation.

1. Case Com. Then $P \equiv a\,!\,l_i[\tilde{v}] \mid a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}$, $Q \equiv \{\tilde{v}/\tilde{x}_i\}P_i$, and $P \xrightarrow{a} Q$. The result of counting the number of times that X is called in P regarding D is

$$\begin{aligned}\mathcal{U}(X, D, P) &= \mathcal{U}(X, D, a\,!\,l_i[\tilde{v}]) + \\ &\qquad \mathcal{U}(X, D, a\,?\,\{l_1\,(\tilde{x}_1) = P_1, \ldots, l_n\,(\tilde{x}_n) = P_n\}) \\ &= \mathcal{U}(X, D, P_1) \sqcup \cdots \sqcup \mathcal{U}(X, D, P_n) \\ &\geq \mathcal{U}(X, D, P_i) = \mathcal{U}(X, D, \{\tilde{v}/\tilde{x}_i\}P_i)\end{aligned}$$

by definition of $\sqcup$. Notice that name substitution has no effect on process variables, hence $\mathcal{U}(X, D, P_i) = \mathcal{U}(X, D, \{\tilde{v}/\tilde{x}_i\}P_i)$.

2. Case PAR. Then $P \equiv P_1 \mid P_2$, $P_1 \xrightarrow{\ell} P_1'$, and $Q \equiv P_1' \mid P_2$. The application of function $\mathcal{U}(X, D, P)$ yields $\mathcal{U}(X, D, P_1) + \mathcal{U}(X, D, P_2)$. By induction hypothesis, $\mathcal{U}(U, D, P_1) \geq \mathcal{U}(U, D, P_1')$. Then

$$\begin{aligned}\mathcal{U}(X, D, P) &= \mathcal{U}(X, D, P_1) + \mathcal{U}(X, D, P_2) \\ &\geq \mathcal{U}(X, D, P_1') + \mathcal{U}(X, D, P_2) = \mathcal{U}(X, D, Q)\end{aligned}$$

3. Case RES$_1$. Then $P \equiv \mathsf{new}\ x : \alpha^{(\kappa_1, \kappa_2)} P_1$, $Q \equiv \mathsf{new}\ x : \alpha^{(\kappa^-, \kappa^-)} P_1'$, $P_1 \xrightarrow{x} P_1'$, and $P \xrightarrow{\epsilon} Q$. Hence,

$$\begin{aligned}\mathcal{U}(X, D, Q) &= \mathcal{U}(X, D, \mathsf{new}\ x : \alpha^{(\kappa_1, \kappa_2)} P_1) \\ &= \mathcal{U}(X, D, P_1)\end{aligned}$$

that, by induction hypothesis,

$$\geq \mathcal{U}(X, D, P_1') = \mathcal{U}(X, D, \mathsf{new}\ x : \alpha^{(\kappa_1^-, \kappa^-)} P_1')$$

4. Case RES$_2$. This case is analogous to RES$_1$.
5. Case DEF. Then, $P \equiv \mathsf{def}\ D'\ \mathsf{in}\ P'$ and $Q \equiv \mathsf{def}\ D'\ \mathsf{in}\ Q'$ with $P' \xrightarrow{x} Q'$. Thus,

$$\mathcal{U}(X, D, P) = \mathcal{U}(U, D', P') + \sum_j \{\mathcal{U}(X_j, D', P') \times \mathcal{U}(X, D, P_j)\}$$

by induction hypothesis

$$\begin{aligned}&\geq \mathcal{U}(X, D', Q') + \sum_j \{\mathcal{U}(X_j, D', Q') \times \mathcal{U}(X, D, P_j)\} \\ &= \mathcal{U}(X, D, Q)\end{aligned}$$

6. Case CALL. Then $P \equiv \mathsf{def}\ D'\ \mathsf{in}\ X_i[\tilde{v}] \mid R$ and $Q \equiv \mathsf{def}\, D'\ \mathsf{in}\ \{\tilde{v}/\tilde{x}_i\}P_i \mid R$ with $D \stackrel{\text{def}}{=} X_1(\tilde{x}_1) = P_1\ \mathsf{and}\ \ldots\ \mathsf{and}\ X_n(\tilde{x}_n) = P_n$. Hence,

$$\begin{aligned}\mathcal{U}(X, D, P) = \mathcal{U}(U, D', X_i[\tilde{v}] \mid R) \\ \sum_j \{(\mathcal{U}(X_j, D', X_i[\tilde{v}] \mid R) \times \mathcal{U}(X, D, P_j)\}\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathcal{U}(X, D, Q) = \mathcal{U}(U, D', \{\tilde{v}/\tilde{x}_i\}P_i \mid R) \\ \sum_j \{(\mathcal{U}(X_j, D', \{\tilde{v}/\tilde{x}_i\}P_i \mid R) \times \mathcal{U}(X, D, P_j)\}\end{aligned}$$

By definition of $\mathcal{U}$, $\mathcal{U}(X, D', X_i[\tilde{v}] \mid R) \geq \mathcal{U}(X, D', \{\tilde{v}/\tilde{x}_i\}P_i \mid R)$.

7. Case Str. Then, $P \xrightarrow{\ell} Q$, $P \equiv R$, $S \equiv Q$, and $R \xrightarrow{\ell} S$. Therefore,

$$\begin{aligned}
\mathcal{U}(X, D, P) &= \mathcal{U}(X, D, R), && \text{by lemma 48} \quad (P \equiv R) \\
&\geq \mathcal{U}(X, D, S), && \text{by induction hypothesis} \\
&= \mathcal{U}(X, D, Q), && \text{by lemma 48} \quad (S \equiv Q)
\end{aligned}$$

Proof of the second and third assertions. The proof proceed by case analysis of whether X is or not the same variable as X_i in $\mathcal{U}(X, D, X_i[\tilde{v}]R)$.

That concludes the proof. □