Type Inference for the receptive distributed Pi-calculus

Cédric Lhoussaine

Type Inference for the receptive distributed π-calculus

Cédric Lhoussaine

N° 4373

Janvier 2002

THÈME 1

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4373--FR+ENG

Type Inference for the receptive distributed π-calculus

Cédric Lhoussaine

Thème 1 — Réseaux et systèmes
Projet MIMOSA

Rapport de recherche n° 4373 — Janvier 2002 — 30 pages

Abstract: In this paper we study the type inference problem for an extended version of the type system of $\text{D}\pi_1^r$ very closed to the one of Hennessy and Riely's $\text{D}\pi$. These are distributed π-calculus involving explicit notions of *locations* and *migration* where the location space is *flat* and communication is *local*. Moreover, location names are typed and we use an explicit subtyping relation over location types that enable us to define a notion of principal typing. We provide an inference type algorithm computing a principal type for all typable term.

Key-words: π-calculus, type inference, records

The author works at *Centre de Mathémathiques et d'Informatique* (LIM-CNRS), 39 rue Joliot-Curie, F-13453, Marseille, France. e-mail: lhoussa@cmi.univ-mrs.fr. And is member of *Projet MIMOSA*.

Inférence de types pour le π-calcul réparti réceptif

Résumé : Nous étudions le problème de l'inférence de types pour une version étendue du système de types de $\mathrm{D}\pi_1^r$, un calcul très proche du $\mathrm{D}\pi$ de Hennessy et Riely. Ces calculs sont des π-calculs répartis comprenant des notions explicites de *localités* et de *migration* où l'espace des localités est plat (*i.e.* pas d'imbrication de localités) et la communication est purement locale. De plus, les noms de localités sont typés et on utilise une relation de sous-typage explicite sur les types de localités ce qui nous permet de définir une notion de *typage principal.* On décrit formellement un algorithme produisant un typage principal pour tout terme typable.

Mots-clés : π-calcul, inférence de types, records

Contents

1 Introduction

In [ABL00] we presented a sub-calculus of Hennessy and Riely's *distributed* Dπ ([HR98]). This calculus is based on the polyadic π-calculus, involving explicit notions of *locality* and *migration*. In this model, the distribution is one dimensional: in contrast to the join calculus ([FGL$^+$96]) or the Ambients ([CG98]), locations are not hierarchical in the sense that locations do not contain sub-locations. This kind of distribution (also adopted in [Ama97]) is said *flat*. Moreover, communication is purely local, that is processes have to be co-located in order to be able to communicate. So, messages to remote resources must be explicitly routed.

In [ABL00] we addressed the problem of how to avoid local deadlocks. Or, more precisely, how to avoid the situation where a message migrates to, or stays in a locality where no receiver will ever be available. The property that avoids this situation was called *message deliverability* and is essentially captured by means of an inference system proving relations $I \Vdash P$ where I is a set of variables that is the set of names on which P persistently provides a receiver. We shown that this relation is not enough to guarantee the message deliverability; indeed processes have to be well typed. The type system we used is a simplified version of the simple type system of [HR98]; that is we use location types ($\{a : \gamma_1, \ldots, a : \gamma_n\}$) that records the names and types that may be used to communicate inside a locality, and we use "located types" ($\gamma^@$) for channels that are sent together with their location instead of the more expressive existential types of [HR98]. More precisely, located types $\gamma^@$ are used for typing "compound names" of the form $x@\ell$ meaning that x is a channel with type γ at location ℓ. Whereas, with existential types $\gamma@\psi$ for a compound name $x@\ell$: x has type γ at location ℓ of type ψ. Existential types permit to give additional informations on the location being transmitted together with a channel name. The use of location types allows to type channels communicating location names. For instance, consider the process $\overline{a}\ell \mid a(k).P$ that sends a location name ℓ on channel a and binds ℓ to k in P. In such a process the name a should have a type *Ch*(ψ) (*i.e.* the type of channels communicating locations of type ψ). Then P is allowed to use channels at location ℓ that are defined (typed) in ψ and *only those ones*. However, it seems natural to allow the location name sent to have a more generous type than these intended by a for k. That is, for instance, if $\psi = \{b : \gamma_1\}$ and ℓ has type $\{b : \gamma_1, c : \gamma_2\}$. In [ABL00], allowing this typing is achieved by means of implicit subtyping on location types, whereas in [HR98] this subtyping is explicit.

The aim of this paper is to describe a type inference algorithm for an extended version of the type system given in [ABL00]. The main extension is the use of existential types $\tau@\psi$ of [HR98]. By this way, we believe that this algorithm should be easily adapted to the simple type system (without read/write capabilities) of [HR98].

In general, when we are seeking for a type inference algorithm, we expect, given a term, that it provides a most general typing. That is a "representative" of all the other possible typings. More precisely, all the typing for a term must be an instance of its principal typing (by substitution of type variables). And, conversely, all instance of the principal typing is a valid typing. However, this classical definition is not sufficient in presence of location types, and more specifically with location subtyping. Therefore, in order to retrieve a notion of "representative" typing, we make subtyping explicit and make use of location type variables (also called *row variables* in the terminology of record types of [Wan87]). Therefore we are able to define principal typing together with a set of subtyping assertions that have to be preserved by type variable instantiation.

This paper is organized as follows. In section 2, we detail the calculus we are considering and we describe the types. Then, in section 3, we present the type system illustrated by several examples and we define the notion of principal typing. In section 4, we deal with the algorithm of unification computing the substitution solution of a set of equations between types and inequations between location types. Next, in section 5, we give the inference algorithm and the main theorems of this paper. Finally, in the last section, we conclude with a summary of our results and open issues.

2 A calculus with localities

In this section we introduce the distributed calculus we will consider along this paper, we give the syntax of types and the inference system. This calculus is a monadic version of the distributed π-calculus given in [ABL00]. Here, by monadic, we mean communications of *one* object, actual and formal parameters of recursions being also submitted to this form of monadicity. Actually, we would have consider the full polyadic version but we made the choice of monadicity for the sake of simplicity. Apart from that this calculus is the usual asynchronous π-calculus with some primitives for spacial distribution and migration of processes organized as a two-levels model: the processes (or thread) one and the network (or configuration) one. As in [HR98] communication is local, that is we cannot directly send a message from a location ℓ to a process located at a different location ℓ': we first have to migrate the message to ℓ', and then we can perform the communication.

$u, v \ldots$	$::=$		*names*
		a	*channel*
	$\mid$	p	*value*
	$\mid$	ℓ	*location*
	$\mid$	$a@\ell$	*compound name*
$P, Q, R \ldots$	$::=$		*processes*
		$\mathbf{0}$	*inaction*
	$\mid$	$\overline{a}u$	*message*
	$\mid$	$a(u).P$	*input*
	$\mid$	$(P \mid Q)$	*parallel composition*
	$\mid$	$[a = b]P, Q$	*conditional branching*
	$\mid$	$(\nu u)P$	*scoping (restriction)*
	$\mid$	$\mathsf{go}\,\ell.P$	*migration*
	$\mid$	$T(u)$	*parametric process instantiation*
T	$::=$		*parametric processes*
		A	*identifi er*
	$\mid$	$(\mathsf{rec}\,A(u).P)$	*recursive parametric process*
S	$::=$		*networks*
		$\mathbf{0}$	*empty network*
	$\mid$	$[\ell :: P]$	*located process*
	$\mid$	$(S \mid S')$	*parallel composition*
	$\mid$	$(\nu a@\ell)S \mid (\nu\ell)S \mid (\nu p)S$	*scoping (restriction)*

Figure 1: Syntax of terms

In order to state the syntax, we consider a denumerable set $\mathcal{N}$ of (*simple*) *names*, which is assumed partitioned into three sets: the channel names $\mathcal{N}_{chan}$ (ranged over by $a, b, c, \ldots$) , the location names $\mathcal{N}_{loc}$ (ranged over by $\ell, k, \ldots$), and the value names $\mathcal{N}_{val}$ (ranged over by $p, q, r, \ldots$). We also set $\mathcal{P}$ to be a denumerable set of process identifiers (ranged over by $A, B, \ldots$) disjoint from $\mathcal{N}$ that will be used for explicit recursive definition instead of replication (see [Mil91, Tur95]). The objects transmitted along channel names may be *compound* that is a channel name a together with a location name ℓ denoted by $a@\ell$ and meaning "a has to be used at location ℓ". The abstraction of a location name is possible at the network level and it requires to provide its type. The grammar of terms can be found in the figure 1. We denote by $U, V \ldots$ a term of any kind – process, parametric process or network. The operational semantics, given in the "chemical style" of Berry and Boudol [BB90], is reported in the appendix B.

We now define types and type system. We assume the existence of a denumerable set of type variables $\mathcal{V}$ partitioned into three sets: the set of *general type variables* $\mathcal{V}_{type}$ ranged over by $t, t', \ldots$, the set of *channel type variables* ranged over by $h, h', \ldots$, and the set of location type variables (usually called *row variables*) ranged over by $\rho_L, \rho'_L, \ldots$ where $L \in \mathcal{P}_{fin}(\mathcal{N}_{chan})$. Types are based on the following grammar:

types	$\tau, \sigma, \ldots$	$::=$	$\psi \mid \gamma \mid \gamma@\psi \mid \mathit{val} \mid t$
channel types	γ	$::=$	$\mathit{Ch}(\tau) \mid h$
location types	ψ	$::=$	$\{a : \gamma, \psi\} \mid \rho_L$

We denote by $\mathit{var}(\tau)$ the set of the type variables occuring in τ. We will often note $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ for $\{a_1 : \gamma_1, \{\ldots, \{a_n : \gamma_n, \rho_L\}\} \ldots\}$. In [ABL00] we stated that typing a location name should mean: "recording the names and types of channels on which communication is possible inside the location". Therefore, a location type is just a record of channel names together with their types. Extension of location type is achieved by means of a row variable that may be substituted with a location type. The row variable of a location type is obtained by means of the function *ρvar* (formally defined by $\rho var(\rho_L) = \rho_L$ and $\rho var(\{a : \gamma, \psi\}) = \rho var(\psi)$). We also denote the set of names defined in a location type ψ by $chan(\psi)$ (formally defined by $chan(\rho_L) = \emptyset$ and $chan(\{a : \gamma, \psi\}) = \{a\} \cup chan(\psi)$). As for record types we have to be careful with multiply-defined names in location types; in other words in $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ the names $a_1, \ldots, a_n$ have to be distincts. To this aim, we equip the row variable of a type ψ with a subscript, that is a set of names which is meant to contain at least $chan(\psi)$. Location types ψ have to satisfy judgments of form[1] $\psi :: L$ (with $L \in \mathcal{P}_{fin}(\mathcal{N}_{chan})$) defined by:

$$\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_{L'}\} :: L \Leftrightarrow \begin{cases} a_1, \ldots, a_n \text{ all distincts,} \\ L' = \{a_1, \ldots, a_n\} \cup L \\ \{a_1, \ldots, a_n\} \cap L = \emptyset \end{cases}$$

We will say that a location type is *well formed* if it satisfies $\psi :: L$ for some L. And we extend naturally this notion to types. From now all types have to be well formed. For instance, the type $\{a : \gamma, b : \gamma', \rho_{\{a,b\}}\}$ is well formed because it satisfies $\{a : \gamma, b : \gamma', \rho_{\{a,b\}}\} :: \emptyset$. This type can be extended with $\{c : \gamma'', \rho'_{\{a,b,c\}}\}$ (that is $\rho_{\{a,b\}}$ can be substituted by $\{c : \gamma'', \rho'_{\{a,b,c\}}\}$) because $\{a : \gamma, b : \gamma', c : \gamma'', \rho'_{\{a,b,c\}}\} :: \emptyset$. Intuitively, the subscript L of a row variable ρ_L allows one to substitute it for a location type defining channels that do not occur in L, thus avoiding to obtain multiply-defined channel names.

The type of a compound name $a@\ell$ is a "located channel type" $\gamma@\psi$ meaning: "a has the type γ at a remote location". This informal definition will become clearer in the next section. We also have the constant type *val* for names of $\mathcal{N}_{val}$ that may be tested only for equality (not used for communication or as a location name). Moreover, we also allow location names to be tested for equality.

Finally, as in [ABL00] we impose some conditions for terms to be legal :

(i) in an input $a(u).P$, $u \neq a$.
(ii) we make the standard assumption that recursion is *guarded*, that is in $(\mathsf{rec}\, A(u).P)(v)$ all recursive calls to A in P occur under an input guard.
(iii) in location types $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ we assume that $\rho_{L'} \notin var(\gamma_1, \ldots, \gamma_n)$ for some L'.

3 The Type System

In this section we describe a modified version of the type system of [ABL00]. The main modifications of this type system are:

1. the extension of located types $\gamma^{@}$ to existential types $\gamma@\psi$.
2. the elimination of the explicit weakening rule of [ABL00] by taking back weakenings in axioms.
3. taking into account type variables.
4. the use of a trivial subtyping relation between location types.

In [ABL00], in the body P of an input process $a(b@\ell).P$, we were able to use b at location ℓ and only b. For instance, the term $a(b@\ell).\mathsf{go}\,\ell.(\bar{b}v \mid \bar{c}w)$ was not typable because we use c at location ℓ whereas only b is permitted to be used at ℓ. The extension to the types $\gamma@\psi$ enables us to type this term: γ is a type for the bound channel (here b) whereas ψ records types for additional unbound channels (here c). Those types are called *existential types* because $\gamma@\psi$ should be read as $\exists b.\{b : \gamma, \psi\}$. The motivation for the second modification is quite intuitive: a weakening rule can be applied anywhere in the inference tree since it does not depend of the structure of the term being infered. Then it carries a lot of non determinism that is problematic for a type inference algorithm. That why we reformulate the inference system by taking back all the weakenings to the axioms. This allows us to keep the whole context along the inference tree (regarding it bottom-up) while it remains possible to extend it with bound names. The third extension is very standard and it leads to formulation of a type inference algorithm. Considering type schemes, for instance, enlightens the possible extensions of locations by explicit substitutions of row variables. The last modification, that is also nothing but a reformulation of the previous type system, is probably the most important one since it allows us to define a notion of principal type. In fact, whenever we are interested in providing a type inference algorithm it is desirable that, given a term, this algorithm will produce a most general type for each free name occurring in that term, that is whenever a substitution is applied to those most general types, the types obtained are still valid ones for the term. Unfortunately, this definition of principal type (though intuitive) is not sufficient for our purpose.

[1]We assume that the notation :: for these judgments does not confuse with the location constructor of the networks $[\ell :: P]$.

Example 3.1 For instance, consider the network

$$S = [\ell :: \overline{a}\ell \mid \overline{b}p \mid \overline{c}p]$$

where p is a value. Then ℓ has type $\{a : Ch(\psi), b : Ch(val), c : Ch(val), \rho_{\{a,b,c\}}\}$ where ψ can be either $\{b : Ch(val), c : Ch(val), \rho'_{\{b,c\}}\}$, $\{b : Ch(val), \rho'_{\{b\}}\}$, $\{c : Ch(val), \rho'_{\{c\}}\}$ or $\rho'_{\emptyset}$. One might expect $\{a : Ch(\rho'_{\emptyset}), b : Ch(val), c : Ch(val), \rho_{\{a,b,c\}}\}$ to be most general type for ℓ. Obviously this is not the case since, for instance, the application of the substitution $[\{b : Ch(Ch(val)), \rho''_{\{b\}}\}/\rho'_{\emptyset}]$ does not give a valid type for ℓ where b is used as a channel of type $Ch(Ch(val))$ whereas a actually sends ℓ where b has the type $Ch(val)$.

□

We introduce a relation $\psi \sqsubseteq \psi'$ on location types (similar to the one of [HR98]) meaning intuitively: "ψ' defines channel names that are also defined in ψ with the same types". It can be seen as a reverse inclusion relation. More formally,

Definition 3.2 *We define the relation $\sqsubseteq$ on location types as follows:*

- $\psi \sqsubseteq \rho_L$
- $\{a : \tau, \psi\} \sqsubseteq \{a : \tau, \psi'\}$ *if* $\psi \sqsubseteq \psi'$

This relation defines subtyping assertions*. The subtyping assertions of the form $\psi \sqsubseteq \rho_L$ are called* atomic*. Moreover, we note $\psi \equiv \psi'$ if $\psi \sqsubseteq \psi'$ and $\psi' \sqsubseteq \psi$.*

As pointed out in [PS93] and [HR98] by subtyping arguments, the communication is asymmetric. In communication of location names this means that the input constraints the type of the channel subject of the communication. For instance, if the body P of the process $a(\ell).P$ uses channels $a_1, \ldots, a_n$ at location ℓ then a must have a type $Ch(\psi)$ where ψ provides at least a type for each a_i. Therefore, in P, ℓ has a type ψ' such that $\psi \sqsubseteq \psi'$. Symmetrically, when outputting a location k on a, the type ψ'' of k must provide a type for each a_i that are the same as those provided by ψ; that is $\psi'' \sqsubseteq \psi$.

Example 3.3 Consider the following term:

$$S = [\ell :: \overline{a}k \mid a(\ell').\mathsf{go}\,\ell'.\overline{b}p] \mid [k :: \overline{c}p]$$

where p is a value. After an input on a of the location name k we migrate to this location a message on b. Then any location transmited on a has to define at least a channel b with type $Ch(val)$. Then, S can be typed with the following types for ℓ and k:

$$\ell : \{a : Ch(\{b : Ch(val), \rho_{\{b\}}\}), \rho'_{\{a\}}\},\ k : \{b : Ch(val), c : Ch(val), \rho''_{\{b,c\}}\}$$

The channel a may also have the type $Ch(\{b : Ch(val), c : Ch(val), \rho'''_{\{b,c\}}\})$ that can be obtained from the previous type for a by the substitution $[\{c : Ch(val), \rho'''_{\{b,c\}}\}/\rho_{\{b\}}]$. Actually, not all substitutions of $\rho_{\{b\}}$ will give a valid typing for a (as for instance $[\{c : Ch(Ch(val)), \rho_{\{b,c\}}\}/\rho_{\{b\}}]$ or $[\{e : Ch(val), \rho_{\{e,c\}}\}/\rho_{\{b\}}]$). However, making use of atomic subtyping assertions, we are able to trivially decide whether or not a substitution applying to the previous types for ℓ and k gives rise to valid types for S. Indeed, any substitution μ preserving the atomic subtyping assertion $\{c : Ch(val), \rho''_{\{b,c\}}\} \sqsubseteq \rho_{\{b\}}$ (that is such that $\mu\{c : Ch(val), \rho''_{\{b,c\}}\} \sqsubseteq \mu\rho_{\{b\}}$ is still valid), preserves the typing of S.
□

The type system deals with sequents of the form $\Gamma \vdash_\ell P$, for checking that the process P, placed at the current location ℓ, conforms to the typing assumption Γ. We also have sequents of the form $\Gamma \vdash S$ and $\Gamma \vdash_\ell T : \gamma$, respectively for networks and parametric processes. The system also uses two simple auxiliary systems for typing names, whose sequents have the form $\Gamma \vdash_\ell u : \tau$ and $\Gamma \vdash^W_\ell u : \tau$, the second one embedding weakenings. In these sequents the typing context Γ is a mapping from a finite subset $dom(\Gamma)$ of $\mathcal{N}_{val} \cup \mathcal{N}_{loc} \cup \mathcal{P}$ into the set of types, subject to the following requirement:

(i) if $p \in dom(\Gamma) \cap \mathcal{N}_{val}$ then $\Gamma(p)$ is *val*.
(ii) if $\ell \in dom(\Gamma) \cap \mathcal{N}_{loc}$ then $\Gamma(\ell)$ is a location type.
(iii) if $A \in dom(\Gamma) \cap \mathcal{P}$ then $\Gamma(A)$ is a channel type.

(iv) if $p, \ell, A \in \mathit{dom}(\Gamma)$ then p, ℓ, A do not occur in $\mathit{im}(\Gamma)$, that is, p, ℓ, A do not occur in any type assigned by Γ.
(v) if $\ell \neq k$ then $\rho\mathit{var}(\Gamma(\ell)) \notin \mathit{var}(\Gamma(k))$

We will write a typing context as usual, that is as a list of typing assumptions:

$$\Gamma = \ldots, p : \mathit{val}, \ldots, \ell : \psi, \ldots, A : \gamma, \ldots$$

We make use of a partial operator of union Δ, Γ of typing contexts defined, if $\mathit{dom}(\Delta) \cap \mathit{dom}(\Gamma) = \emptyset$, as follows:

$$(\Delta, \Gamma)(x) = \begin{cases} \Delta(x) & \text{if } x \in \mathit{dom}(\Delta) \\ \Gamma(x) & \text{if } x \in \mathit{dom}(\Gamma) \end{cases}$$

where x may be a value, a location or an identifier.

In the type system we assume that the contexts satisfy the previous required conditions. We make the usual convention in the rules for binding constructs, that is input, restriction and recursion, that the bound names do not occur free in the resulting sequent. Moreover, we make use of the notation $\psi \sqsubseteq \{a : \gamma\}$ if there is some $\rho_{\{a\}}$ such that $\psi \sqsubseteq \{a : \gamma, \rho_{\{a\}}\}$.

Let us comment the rules of the type system given in figures 2 to 5. In the type system of [ABL00] an implicit form of subtyping is involved in output of localities: the judgment

$$\ell : \{b : \gamma_0, c : \gamma_1\}, k : \{a : \mathit{Ch}(\{\})\} \vdash_k \overline{a}\ell$$

was valid, even though the type of ℓ given by the context is more generous than the one carried by the channel a. In our system this judgment is equivalent (adding row variables) to:

$$\ell : \{b : \gamma_0, c : \gamma_1, \rho_{\{b,c\}}\}, k : \{a : \mathit{Ch}(\rho'_\emptyset), \rho''_{\{a\}}\} \vdash_k \overline{a}\ell$$

that can be derived with the explicit subtyping relation involved in the weak type system for names (since $\{b : \gamma_0, c : \gamma_1, \rho_{\{b,c\}}\} \sqsubseteq \rho'_\emptyset$), in place of the weakening rule of [ABL00]. Intuitively, in the weak type system for names we derive judgments $\Gamma \vdash^W_\ell u : \tau$ where Γ provides possibly more than what is necessary for the typing of $u : \tau$ at location ℓ. We make use of the weak type system for names whenever we have to type actual parameters, that is in the output rule and in the typing of the application of a parametric process to arguments.

In the case of formal parameters and more generally whenever we have a binding construct, weakening is obviously not allowed. Therefore, to type the body P of the binding constructs $a(b).P$, $(\nu b)P$, $(\nu b@\ell)P$, $(\nu b@\ell)S$ and $(\mathsf{rec}\, A(b).P)$ we have to provide a type for the bound name b in the context. This is simply done by extending the appropriate location with a type for b. For the other binding constructs (that is $a(u).P$ and $(\mathsf{rec}\, A(u).P)$) we make use of the strong type system for names. In this system we derive judgments $\Gamma \vdash_\ell u : \tau$ where Γ provides exactly what is necessary to type $u : \tau$ at location ℓ and only that.

In the rules for conditional branching, the side conditions tell that we are allowed to test only both names of values or of locations, but not channel names. To type a migrating process $\mathsf{go}\, \ell.P$, one must be able to type P at location ℓ, while the resulting current location is immaterial.

Example 3.4 In this example we show that the process $a(b@\ell).\mathsf{go}\, \ell.(\overline{b}p \mid \overline{c}q)$ discussed at the beginning of this section is typed at a location k in the context

$$\Gamma = k : \{a : \mathit{Ch}(\gamma@\psi), \rho_{\{a\}}\},\ p : \mathit{val},\ q : \mathit{val}$$

where $\gamma = \mathit{Ch}(\mathit{val})$ and $\psi = \{c : \gamma, \rho'_{\{c\}}\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, p : \mathit{val}, q : \mathit{val} \vdash^W_\ell p, q : \mathit{val}}}{\ldots, \ell : \{b : \gamma, c : \gamma, \rho''_{\{b,c\}}\}, \ldots \vdash_\ell (\overline{b}p \mid \overline{c}q)}}{\ldots, \ell : \{b : \gamma, c : \gamma, \rho''_{\{b,c\}}\}, \ldots \vdash_k \mathsf{go}\, \ell.(\overline{b}p \mid \overline{c}q)}}{k : \{a : \mathit{Ch}(\gamma@\psi), \rho_{\{a\}}\}, \ell : \{b : \gamma, c : \gamma, \rho''_{\{b,c\}}\}, p : \mathit{val}, q : \mathit{val} \vdash_k \mathsf{go}\, \ell.(\overline{b}p \mid \overline{c}q)}}{\Gamma \vdash_k a(b@\ell).\mathsf{go}\, \ell.(\overline{b}p \mid \overline{c}q)}$$

$$\frac{}{p : val \vdash_\ell p : val} \qquad \frac{\psi \equiv \psi'}{k : \psi' \vdash_\ell k : \psi} \qquad \frac{\psi' \equiv \psi}{k : \{a : \gamma, \psi'\} \vdash_\ell a@k : \gamma@\psi}$$

Figure 2: Strong type system for names

$$\frac{\Gamma(p) = val}{\Gamma \vdash^W_\ell p : val} \qquad \frac{\Gamma(k) \sqsubseteq \psi}{\Gamma \vdash^W_\ell k : \psi} \qquad \frac{\Gamma(\ell) \sqsubseteq \{a : \gamma\}}{\Gamma \vdash^W_\ell a : \gamma} \qquad \frac{\Gamma(k) \sqsubseteq \{a : \gamma, \psi'\}\ ,\ \psi' \equiv \psi}{\Gamma \vdash^W_\ell a@k : \gamma@\psi}$$

Figure 3: Weak type system for names

$$\frac{}{\Gamma \vdash_\ell \mathbf{0}} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash_k \mathsf{go}\,\ell.P} \qquad \frac{\Gamma \vdash^W_\ell u : \tau\ ,\ \Gamma(\ell) \sqsubseteq \{a : Ch(\tau)\}}{\Gamma \vdash_\ell \overline{a}u}$$

$$\frac{\Delta \vdash_\ell P\ ,\ \Gamma(\ell) \sqsubseteq \{b : Ch(\gamma)\}}{\Gamma \vdash_\ell a(b).P}(*) \qquad \frac{\Gamma, \Delta \vdash_\ell P\ ,\ \Delta \vdash_\ell u : \tau\ ,\ \Gamma(\ell) \sqsubseteq \{a : Ch(\tau)\}}{\Gamma \vdash_\ell a(u).P}$$

$$\frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q \quad , \quad \Gamma(\ell_1), \Gamma(\ell_2) \sqsubseteq \rho_\emptyset}{\Gamma \vdash_\ell [\ell_1 = \ell_2]P\ ,\ Q} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q \quad , \quad \Gamma(p) = \Gamma(q) = val}{\Gamma \vdash_\ell [p = q]P\ ,\ Q}$$

$$\frac{p : val, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu p)P} \qquad \frac{\Delta \vdash_\ell P}{\Gamma \vdash_\ell (\nu b)P}(*) \qquad \frac{\Delta \vdash_k P}{\Gamma \vdash_k (\nu b@\ell)P}(*)$$

$$\frac{\Gamma \vdash_\ell P\ ,\ \Gamma \vdash_\ell Q}{\Gamma \vdash_\ell P \mid Q} \qquad \frac{\Gamma \vdash_\ell T : Ch(\tau)\ ,\ \Gamma \vdash^W_\ell u : \tau}{\Gamma \vdash_\ell T(u)} \qquad \frac{\Gamma(A) = Ch(\tau)\ ,\ \Gamma \vdash^W_\ell u : \tau}{\Gamma \vdash_\ell A(u)}$$

$$\frac{A : Ch(\tau), \Gamma, \Delta \vdash_\ell P\ ,\ \Delta \vdash_\ell u : \tau}{\Gamma \vdash_\ell (\mathsf{rec}\, A(u).P) : Ch(\tau)} \qquad \frac{A : Ch(\tau), \Delta \vdash_\ell P}{\Gamma \vdash_\ell (\mathsf{rec}\, A(b).P) : Ch(\tau)}(*)$$

Figure 4: Type system for processes

$$\frac{}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash [\ell :: P]} \qquad \frac{\Gamma \vdash S\ ,\ \Gamma \vdash S'}{\Gamma \vdash (S \mid S')} \qquad \frac{p : val, \Gamma \vdash S}{\Gamma \vdash (\nu p)S} \qquad \frac{\ell : \psi, \Gamma \vdash S}{\Gamma \vdash (\nu \ell)S} \qquad \frac{\Delta \vdash S}{\Gamma \vdash (\nu b@\ell)S}(*)$$

Figure 5: Type system for networks

(1) with $\Delta = \Gamma$ if $b \notin fn(P)$ (and $b \notin fn(S)$ for the network case); otherwise $\Delta =_{dom(\Gamma)-\ell} \Gamma$, $\Delta(\ell) = [\{b : \gamma, \rho'_{L \cup \{b\}}\}/\rho_L]\Gamma(\ell)$ where $\rho_L = \rho var(\Gamma(\ell))$.

with

$$\frac{\{c : \gamma, \rho''_{\{b,c\}}\} \equiv \psi}{\ell : \{b : \gamma, c : \gamma, \rho''_{\{b,c\}}\} \vdash_k b@\ell : \gamma@\psi}$$

□

In order to establish the definition of principal typing we have to deal with substitutions. A substitution (denoted μ, $\lambda, \ldots$) is, as usual, a finite mapping from type variables to types. For

$$\mu = [\tau_1/t_1, \ldots, \tau_n/t_n]$$

we denote $\mathit{dom}(\mu)$ the set $\{t_1, \ldots, t_n\}$ and $\mathit{vrang}(\mu)$ the set $\bigcup_{i\in\{1\ldots n\}} \mathit{var}(\tau_i)$. All the substitutions we consider are supposed to be idempotent, that is $\mathit{dom}(\mu) \cap \mathit{vrang}(\mu) = \emptyset$. We also make the assumption that substitutions respect the grammar of types: for instance the substitution of a channel variable is a channel type. Moreover, for all $\rho_L \in \mathit{dom}(\mu)$, $\psi = \mu\rho_L$ and $\rho'_{L'} = \rho\mathit{var}(\psi)$, the condition $L \subseteq L'$ must be satisfied. Outside its domain a substitution is supposed to be the identity. Substitutions are trivially extended to homomorphisms on types. The extension of substitutions to other objects (as sequents, subtyping assertions, etc.) is left informal since it is completely standard. We note $\lambda\mu$ the composition of the substitutions λ and μ, and $\emptyset$ the empty substitution.

Definition 3.5 *Let μ be a substitution and A be a set of atomic subtyping assertions, μ is a A-*substitution *if it preserves the assertions of A.*

Definition 3.6 (principal typing) *Let A be a set of atomic subtyping assertions and Γ be a typing context, then $A; \Gamma$ is said to be a* principal typing *for a system S (resp. a process P at ℓ) if and only if*

1. *For all A-substitution μ, we have $\mu\Gamma \vdash S$ (resp. $\mu\Gamma \vdash_\ell P$).*
2. *For all Γ' such that $\Gamma' \vdash S$ (resp. $\Gamma' \vdash_\ell P$), there exists a A-substitution μ such that $\mu\Gamma =_{\mathit{dom}(\Gamma)} \Gamma'$.*

The existence of a principal typing for each typable term is crucial whenever we aim at giving a type inference algorithm. However, without the principal typing property, one could devise an algorithm computing *some* type for any typable term, but this would be quite weak. Principal typing enables us to check that several pieces of code agree in the use of the names they share.

Example 3.7 The term $S = [\ell_1 :: \bar{b}p] \mid [\ell_2 :: \bar{c}d] \mid [k :: \bar{a}\ell_1 \mid \bar{a}\ell_2]$ has principal typing $\Gamma; A$ where

$$\Gamma = \ell_1 : \{b : \mathit{Ch}(\mathit{val}), \rho^1_{\{b\}}\}\ ,\ \ell_2 : \{c : \mathit{Ch}(h), d : h, \rho^2_{\{c,d\}}\}\ ,\ k : \{a : \mathit{Ch}(\rho_\emptyset), \rho'_{\{a\}}\}\ ,\ p : \mathit{val}$$

$$A = \{\{b : \mathit{Ch}(\mathit{val}), \rho^1_{\{b\}}\} \sqsubseteq \rho_\emptyset\ ,\ \{c : \mathit{Ch}(h), d : h, \rho^2_{\{c,d\}}\} \sqsubseteq \rho_\emptyset\}$$

Intuitively, the context Γ tells us that, a is a channel that carries any location name of any type $\rho_\emptyset$. And, A tells that whenever a location name, carried by a, defines a name b (resp. c) it is with the type $\mathit{Ch}(\mathit{val})$ (resp. $\mathit{Ch}(h)$). As long as there is no receiver on a, the type of the location carried by a remains unspecified. We can see that, the system $S' = [k :: a(\ell).\mathsf{go}\ \ell.\bar{b}p]$ can be composed with S because it agrees with the principal type. Indeed, in S' the type of a is $\mathit{Ch}(\{b : \mathit{Ch}(\mathit{val}), \rho^3_{\{b\}}\})$, and in Γ this type is obtained by means of the substitution $[\{b : \mathit{Ch}(\mathit{val}), \rho^3_{\{b\}}\}/\rho_\emptyset]$. However, this is not sufficient for a valid typing since the type of ℓ_2 does not then agree with the type of a. This can also be noted by the fact that this substitution is not a A-substitution, that is it preserves the validity of the assertion $\{b : \mathit{Ch}(\mathit{val}), \rho^1_{\{b\}}\} \sqsubseteq \rho_\emptyset$ but not the one of $\{c : \mathit{Ch}(h), d : h, \rho^2_{\{c,d\}}\} \sqsubseteq \rho_\emptyset$. Then, a correct A-substitution is, for instance,

$$[\ \{b : \mathit{Ch}(\mathit{val}), \rho^3_{\{b\}}\}/\rho_\emptyset\ ,\ \{b : \mathit{Ch}(\mathit{val}), \rho^4_{\{b,c\}}\}/\rho^2_{\{c,d\}}\ ]$$

□

In the next example we see how the set of subtyping assertions can forbid the use of a channel of a transmited location name.

Example 3.8 Consider the following term which is a slight modification of the one of the previous example:

$$S'' = [\ell_1 :: \overline{b}p] \mid [\ell_2 :: \overline{b}d] \mid [k :: \overline{a}\ell_1 \mid \overline{a}\ell_2]$$

where p is a value and d is a channel. Although b is used with two different types, this term remains typable since the two uses of b are at two different locations. A principal typing is then

$$\Gamma = \ell_1 : \{b : Ch(val), \rho^1_{\{b\}}\} , \ \ell_2 : \{b : Ch(h), d : h, \rho^2_{\{b,d\}}\} , \ k : \{a : Ch(\rho_\emptyset), \rho'_{\{a\}}\} , \ p : val$$

$$A = \{\{b : Ch(val), \rho^1_{\{b\}}\} \sqsubseteq \rho_\emptyset , \ \{b : Ch(h), d : h, \rho^2_{\{c,b\}}\} \sqsubseteq \rho_\emptyset\}$$

Now, we are no longer able to compose S' of example 3.7 with S'' because it is easy to see that there exists no A-substitution (that is, such that it validates both $\{b : Ch(val), \rho^1_{\{b\}}\} \sqsubseteq \rho_\emptyset$ and $\{b : Ch(h), d : h, \rho^2_{\{c,b\}}\} \sqsubseteq \rho_\emptyset$). □

In the example 3.7 we implicitly made unification of types (the type of a in S with its type in S') in order to find a typing for the composition $(S \mid S')$. In other words, we infered a typing for $(S \mid S')$ from those of S and S'. As well known, unification is the basis of type inference. This is the subject of the next section.

4 Unifi cation

In this section we give a unification algorithm in terms of a rewriting system. The usual problem of unification is: given a set of equations between types, does it exist a substitution for type variables that equates types ? As for unification of record types, we use equational unification, that is syntactic equality of terms is replaced by equality modulo an equational theory defined by a set of identities E. This equational theory is the least congruence relation on the type algebra that is closed under substitution and contains E. For our purpose, this set is reduced to a single equation meaning that the ordering of the fields of location types doesn't matter. For instance, $\{a : \gamma, b : \gamma', \rho_{\{a,b\}}\}$ and $\{b : \gamma', a : \gamma, \rho_{\{a,b\}}\}$ define the same location type. Therefore, the equation E is:

$$\{a : h, \{b : h', \rho_L\}\} =_E \{b : h', \{a : h, \rho_L\}\}$$

From now on all type equality is assumed to be modulo E.

Definition 4.1 *A* typing problem *is a set C of equations between types $\{\tau_1 \doteq \sigma_1, \dots, \tau_n \doteq \sigma_n\}$. We say that a substitution μ is a* solution *of C (or a* unifier *for C) if it unifi es all pairs of types in C, that is such that $\mu\tau_i = \mu\sigma_i$ for all $i \in \{1 \dots n\}$.*

In the syntactic theory, following the terminology of [Rob65], whenever two terms α and β are unifiable, there exists a *most general unifi er* μ, that is such that for all solution λ of $\alpha \doteq \beta$, there exists μ' such that $\lambda = \mu'\mu$. In general, in equational theories, a more general substitution does not exist as it is shown in the following example (for instance, see also [BS99] for more on unification theory).

Example 4.2 Let f be a binary commutative function and E_c be the equation

$$f(x, y) =_{E_c} f(y, x)$$

establishing the commutativity of f. In this equational theory, the problem $f(x, y) \doteq f(a, b)$ where a and b are constant symbols, has the two solutions $\mu_1 = [a/x, b/y]$ and $\mu_2 = [a/y, b/x]$. However, there is no μ' such that $\mu_1 = \mu'\mu_2$ or $\mu_2 = \mu'\mu_1$. □

Nevertheless, we can define a quasi-ordering on substitutions. We note $\mu =_{\mathcal{X}} \lambda$ where $\mathcal{X}$ is a set of type variables if $\mu t = \lambda t$ for all $t \in \mathcal{X}$, and we say, as in [JM93], that μ *is equal to* λ *with respect to* $\mathcal{X}$.

Definition 4.3 *Let μ, λ be substitutions and $\mathcal{X}$ a set of type variables. μ is* more general on $\mathcal{X}$ *than λ iff there exists a substitution μ' such that $\lambda =_{\mathcal{X}} \mu'\mu$. In this case we write $\mu \leq_{\mathcal{X}} \lambda$.*

Given a problem C and a set of type variables $\mathcal{X}$ such that $var(C) \subset \mathcal{X}$, we do not then consider a most general solution but a minimal complete set of solutions $\mathcal{S}_C$ of C satisfying:

1. for each λ solution of C, there exists $\mu \in \mathcal{S}_C$ such that $\mu \leq_{\mathcal{X}} \lambda$.

2. two distinct elements of $\mathcal{S}_C$ are incomparable w.r.t. $\leq_{\mathcal{X}}$.

We say that a problem C has type unitary if and only if it has a minimal complete set of solutions of cardinality 1. An equational theory has type unitary if for all problem C, its minimal complete set of solutions has a cardinality of at most 1. In [Rém93b, Rém93a], Rémy stated that unification in the record algebra is decidable and unitary. Since our location types are nothing but record types, this result is also valid for our purpose. Then, all typing problem that has a solution, has a minimal complete set of solutions reduced to a single element. Then, this one can be called a most general solution w.r.t $\mathcal{X}$.

In order to find a most general typing for the terms of our distributed calculus, we also have to deal with subtyping problems whose definition is the following.

Definition 4.4 *A* subtyping problem *is a set A of subtyping inequations between location types*

$$\{\psi_1 \dot{\sqsubseteq} \psi'_1, \ldots, \psi_n \dot{\sqsubseteq} \psi'_n\}$$

We say that a substitution μ is a solution of A if it validates the inequations of A, that is such that $\mu\psi_i \sqsubseteq \mu\psi'_i$ for all $i \in \{1 \ldots n\}$. We use the notation $\psi \dot{\equiv} \psi'$ as aN abbreviation for $\psi \dot{\sqsubseteq} \psi' \wedge \psi' \dot{\sqsubseteq} \psi$.

We call problem a pair (C, A) consisting of a typing problem and a subtyping problem. In the next section we will see that the type inference problem reduces to the problem of finding a most general solution of a problem (C, A).

In figure 6 we define a reduction relation $\rightarrow$ on tuples $(C\ ,\ A\ ,\ \mu)$ where (C, A) is a problem and μ a substitution. We make use of the abbreviation $\psi \triangleleft \rho_L$ for $[\rho_L/\rho\mathit{var}(\psi)]\psi$. The idea is that starting from a problem with the empty substitution $(C\ ,\ A\ ,\ \emptyset)$ we apply the reduction relation until we reach either a configuration $(\emptyset\ ,\ A'\ ,\ \mu)$ where μ is a most general solution for the problem (C, A)[2], or the failure configuration $\perp$ if (C, A) has no solution. This relation is quite usual: it almost consists of the decompositions of pairs of types until one of these is a type variable. Then, the current substitution μ is composed with the substitution of the type variable for the non-variable type (this last one being also applied to the remaining typing and subtyping problems, see rule *(elim)*). The side condition $t \notin \mathit{var}(\tau)$ of rule *(elim)* ensures that there will not remain occurrences of the type variable being eliminated in the resulting problem, thus avoiding possible infinite reductions. If this condition is not verified, the rule *(oc)* (occurs check) applies and leaves us in the failure configuration. The rules *(trivial)* and *(trivial$_{val}$)* remove trivial equations (that is for which all substitution is a solution) from the typing problem. The rules *(chan)* and *(at)* decompose types in an obvious way (since, for instance, all the solutions of $\mathit{Ch}(\tau) \doteq \mathit{Ch}(\sigma)$ are also these of $\tau \doteq \sigma$). The rule *(loc)* says that to unify $\{a : \gamma, \psi\}$ with $\{a : \gamma', \psi'\}$, we have to unify γ with γ' and ψ with ψ'. It is justified by the following lemma that is proved in the appendix.[3]

Lemma 4.5 *The following holds:*

1. *μ is a solution of $\{\{a : \gamma, \psi\} \doteq \{a : \gamma', \psi'\}\} \cup C$ iff it is a solution of $\{\gamma \doteq \gamma', \psi \doteq \psi'\} \cup C$.*

2. *Let ψ and ψ' be location types such that $\mathit{chan}(\psi) \cap \mathit{chan}(\psi') = \emptyset$, $\rho_L = \rho\mathit{var}(\psi)$ and $\rho'_{L'} = \rho\mathit{var}(\psi')$; μ unifies $\{\psi \doteq \psi'\} \cup C$ iff*

   (a) $\rho_L \notin \mathit{var}(\psi') - \rho'_{L'}$ and $\rho'_{L'} \notin \mathit{var}(\psi) - \rho_L$;

   (b) $\mathit{chan}(\psi) \cap L' = \emptyset$ and $\mathit{chan}(\psi') \cap L = \emptyset$;

   (c) for any $\mathcal{X}'$, for any $\rho''_{L \cup L'} \notin \mathcal{X} \cup \mathcal{X}'$ (where $\mathcal{X}$ is the set of type variables occurring in $\{\psi \doteq \psi'\} \cup C$), there exists λ such that $\mu =_{\mathcal{X} \cup \mathcal{X}'} \lambda\mu'$ and λ unifies $\mu' C$ where $\mu' = [\psi \triangleleft \rho''_{L \cup L'}/\rho'_{L'}\ ,\ \psi' \triangleleft \rho''_{L \cup L'}/\rho_L]$.

This lemma also justifies the side conditions of the rule *(loc_end)*. Intuitively, to unify two location types ψ and ψ' that type disjoint sets of channel names (that is, such that $\mathit{chan}(\psi) \cap \mathit{chan}(\psi') = \emptyset$), we have to extend ψ with channels defined in ψ' by substituting its row variable with a location type typing channels as in ψ'. And the converse for extending ψ'. In the rule *clash*, we fall in the failure configuration if two non variable types do not match.

The rules *(st_ok)* and *(st_wrg)* are used to resolve subtyping inequations. The first one asserts that $\{a : \gamma, \psi\} \dot{\sqsubseteq} \{a : \gamma', \psi'\}$ has a solution if we can unify γ with γ' and if $\psi \dot{\sqsubseteq} \psi'$ has a solution. Indeed, a condition for location types

[2]more precisely μ is most general solution for C and a solution for A

[3]A more general version of this lemma is also established in [JM93].

(trivial) $$(\{t \doteq t\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (C\,,\, A\,,\, \mu)$$

(trivial$_{val}$) $$(\{val \doteq val\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (C\,,\, A\,,\, \mu)$$

(elim) $$(\{t \doteq \tau\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; ([\tau/t]C\,,\, [\tau/t]A\,,\, [\tau/t]\mu)$$

if τ is not a location type, $t \notin var(\tau)$, and $t \notin dom(\mu)$

(chan) $$(\{Ch(\tau) \doteq Ch(\sigma)\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (\{\tau \doteq \sigma\} \cup C\,,\, A\,,\, \mu)$$

(at) $$(\{\gamma_1@\psi_1 \doteq \gamma_2@\psi_2\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (\{\gamma_1 \doteq \gamma_2, \psi_1 \doteq \psi_2\} \cup C\,,\, A\,,\, \mu)$$

(loc) $$(\{\{a:\gamma,\psi\} \doteq \{a:\gamma',\psi'\}\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (\{\gamma \doteq \gamma'\} \cup \{\psi \doteq \psi'\} \cup C\,,\, A\,,\, \mu)$$

(loc_end) $$(\{\psi_1 \doteq \psi_2\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; (\lambda C,\, \lambda A\,,\, \lambda\mu)$$

where $\rho_L = \rho var(\psi_1)$ and $\rho'_{L'} = \rho var(\psi_2)$, $\rho''_{L\cup L'}$ fresh row variable and
$\psi'_1 = \psi_1 \triangleleft \rho''_{L\cup L'}$, $\psi'_2 = \psi_2 \triangleleft \rho''_{L\cup L'}$ and $\lambda = [\psi'_1/\rho'_{L'}, \psi'_2/\rho_L]$ and if

- $chan(\psi_1) \cap chan(\psi_2) = \emptyset$
- $L' \cap chan(\psi_1) = \emptyset$ and $L \cap chan(\psi_2) = \emptyset$
- $\rho_L \notin var(\psi_2) - \rho'_{L'}$ and $\rho'_{L'} \notin var(\psi_1) - \rho_L$

(clash) $$(\{\sigma \doteq \tau\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; \bot$$

if σ and τ are not variables and have not the same top symbol, or are location types with disjoint sets of channels but one of the conditions of *(loc_end)* is not satisfied.

(oc) $$(\{t \doteq \tau\} \cup C\,,\, A\,,\, \mu) \;\rightarrow\; \bot$$

if $t \in var(\tau)$.

(st_ok) $$(C\,,\, \{\{a:\gamma,\psi\} \dot{\sqsubseteq} \{a:\gamma',\psi'\}\} \cup A\,,\, \mu) \;\rightarrow\; (\{\gamma \doteq \gamma'\} \cup C\,,\, \{\psi \dot{\sqsubseteq} \psi'\} \cup A\,,\, \mu)$$

(st_wrg) $$(C\,,\, \{\psi_1 \dot{\sqsubseteq} \psi_2\} \cup A\,,\, \mu) \;\rightarrow\; (\lambda C\,,\, \{\psi_1 \triangleleft \rho''_{L\cup L'} \dot{\sqsubseteq} \rho'_{L'}\} \cup \lambda A\,,\, \lambda\mu)$$

where $\rho_L = \rho var(\psi_1)$, $\rho'_{L'} = \rho var(\psi_2)$, $\rho''_{L\cup L'}$ a fresh row variable, ψ_2 is not a row variable,
$\lambda = [\Psi_2 \triangleleft \rho''_{L\cup L'}/\rho_L]$ and if $chan(\psi_1) \cap chan(\psi_2) = \emptyset$, $L \cap chan(\psi_2) = \emptyset$ and $\rho_L \notin var(\psi_2) - \rho'_{L'}$.

Figure 6: Reduction relation for unification

to be in subtyping relation is that they assign the same types to common channel names. The second rule applies on $\psi \dot{\sqsubseteq} \psi'$ when all the channels defined in ψ' are not in ψ. Obviously, ψ and ψ' are not in subtyping relation. Therefore, we have to extend ψ with the channels typed in ψ'.

Now, we give the main results of this section validating our reduction relation. Their proofs are reported to the appendix C. The first one establishes the termination of $\rightarrow$, and the second one the preservation of the solutions along the reductions.

Lemma 4.6 (Termination) *Given a problem (C, A), a sequence of reductions*

$$(C\,,\, A\,,\, \emptyset) \rightarrow (C_1\,,\, A_1\,,\, \mu_1) \rightarrow \ldots$$

terminates either with $\bot$ or with $(\emptyset\,,\, A'\,,\, \mu)$ where A' is a set of atomic subtyping inequations and μ is a substitution.

Lemma 4.7 (Preservation) *If $(C\,,\, A\,,\, \mu) \rightarrow (C'\,,\, A'\,,\, \mu')$ then*

- *either $\mu = \mu'$ and λ is a solution of (C, A) iff it is a solution for (C', A'),*

(n₁)	$(\_\,,\,\_\,,\,\{\Gamma\vdash^W_\ell k:\psi\}\cup J_n\,,\,C\,,\,A)$	$\rightarrow$	$(\_\,,\,\_\,,\,J_n\,,\,C\,,\,A\cup\{\Gamma(k)\,\dot{\sqsubseteq}\,\psi\})$
(n₂)	$(\_\,,\,\_\,,\,\{\Gamma\vdash^W_\ell a:\gamma\}\cup J_n\,,\,C\,,\,\_)$	$\rightarrow$	$(\_\,,\,\_\,,\,J_n\,,\,C\cup\{\Gamma(\ell)\doteq\{a:\gamma,\rho_{\{a\}}\}\}\,,\,\_)$
(n₃)	$(\_\,,\_\,,\{\Gamma\vdash^W_\ell a@k:\tau\}\cup J_n\,,\,C\,,\,A)$	$\rightarrow$	$(\_\,,\,\_\,,\,J_n\,,\,C\cup\{\Gamma(k)\doteq\{a:h,\rho_{\{a\}}\},\ h@\rho'_\emptyset\doteq\tau\},$ $\{\rho_{\{a\}}\,\dot{\sqsubseteq}\,\rho'_\emptyset\}\cup A)$
(n₄)	$(\_\,,\,\_\,,\,\{\Gamma\vdash^W_\ell p:\tau\}\cup J_n\,,\,C\,,\,\_)$	$\rightarrow$	$(\_\,,\,\_\,,\,J_n\,,\,C\cup\{\Gamma(p)\doteq \mathit{val},\tau\doteq \mathit{val}\}\,,\,\_)$
(nc)	$(\_\,,\,\_\,,\,\{\Gamma\vdash^W_\ell u:\tau\}\cup J_n\,,\,\_\,,\,\_)$	$\rightarrow$	$\bot$

if Γ is not well formed.

Figure 7: Generation of constraints for names

- *or* $\mu=\mu''\mu'$ *and* $\lambda=_{var(C,A)}\lambda'\mu''$*, for some* λ *and* λ'*, then* λ *is a solution of* (C,A) *iff* λ' *is a solution for* (C',A')*.*

The following proposition shows that our reduction relation is sound, that is starting from a problem, if the reduction ends on a substitution then it is a most general solution for the initial problem.

Proposition 4.8 (Soundness) *If* $(C\,,\,A\,,\,\emptyset)\rightarrow^*(\emptyset\,,\,A'\,,\,\mu)$ *and* $(\emptyset\,,\,A'\,,\,\mu)$ *does not reduce anymore, then* μ *is a most general solution w.r.t. var*(C,A) *of* C *and is a solution for* (C,A)*.*

Lemma 4.9 $(C\,,\,A\,,\,\emptyset)\rightarrow^*\bot$ *iff* (C,A) *has no solution.*

The last proposition of this section states the completeness of $\rightarrow$.

Proposition 4.10 (Completeness) *If* (C,A) *has a solution, then* $(C\,,\,A\,,\,\emptyset)\rightarrow^*(\emptyset\,,\,A'\,,\,\mu)$ *where* A' *is atomic.*

5 Type Inference

In this section we describe the process of type inference, that is we wish, starting from a network term and a minimal typing context, to generate a problem whose solution applied to the initial context will give a principal typing. By initial context, for a term S, we mean the set of location names and value names occurring free in S, associated with a row variable or *val* as type. Since such a context can be easily computed we do not give the corresponding algorithm. The idea of the type inference is to incrementally build the inference tree of the typing of a term. That is the inference tree in the inference system described in section 3. This is done by means of a rewriting system which acts on tuples

$$(J_s\,,\,J_p\,,\,J_n\,,\,C\,,\,A)$$

where J_s (resp. J_p and J_n) is a set of sequents $\Gamma\vdash S$ for networks (resp. $\Gamma\vdash_\ell P$ for processes and $\Gamma\vdash^W_\ell u:\tau$ for names) and (C,A) is the problem being generated. The reduction is very closed to the inference system. Indeed, given a sequent in the tuple, the reduction will mostly consist in replacing it by the sequents that are premises of the corresponding rule in the inference system. Eventually, constraints will also be added accordingly. For instance, suppose that $J_s=\{\Gamma\vdash S\mid S'\}\cup J'_s$, the rule for typing parallel composition of networks is

$$\frac{\Gamma\vdash S\qquad\Gamma\vdash S'}{\Gamma\vdash S\mid S'}$$

Then the reduction will simply be

$$(\{\Gamma\vdash S\mid S'\}\cup J'_s\,,\,J_p\,,\,J_n\,,\,C\,,\,A)\rightarrow(\{\Gamma\vdash S\,,\,\Gamma\vdash S'\}\cup J'_s\,,\,J_p\,,\,J_n\,,\,C\,,\,A)$$

(p₀) $$(\_, \{\Gamma \vdash_\ell \mathbf{0}\} \cup J_p\,, \_, \_, \_) \rightarrow (\_, J_p\,, \_, \_, \_)$$

(p₁) $$(\_, \{\Gamma \vdash_\ell \overline{a}u\} \cup J_p\,, J_n\,, C\,, \_) \rightarrow (\_, J_p\,, \{\Gamma \vdash^W_\ell u : t\} \cup J_n\,, C \cup C'\,, \_)$$

$C' = \{\Gamma(\ell) \doteq \{a : \mathit{Ch}(t), \rho''_{\{a\}}\}$

(p₂) $$(\_, \{\Gamma \vdash_\ell a(u).P\} \cup J_p\,, \_, C\,, A) \rightarrow (\_, \{\Delta \vdash_\ell P\} \cup J_p, \_, C \cup C' \cup C'', A \cup A')$$

if $u = b$ then $\Delta =_{\mathit{dom}(\Gamma)-\ell} \Gamma,\ \Delta(\ell) = [\{b : h, \rho'_{L\cup\{b\}}\}/\rho_L]\Gamma(\ell),\ \rho_L = \rho\mathit{var}(\Gamma(\ell))$
$C'' = \{t \doteq h, \rho_L \doteq \rho'_{L\cup\{b\}}\},\ A' = \emptyset;$ otherwise $\Delta = \Gamma,\ \Gamma'$ when $\mathit{gen}(u : t, \ell) = (\Gamma', C'', A')$
and $C' = \{\Gamma(\ell) \doteq \{a : \mathit{Ch}(t), \rho''_{\{a\}}\}$

(p₃) $$(\_, \{\Gamma \vdash_\ell P \mid Q\} \cup J_p\,, \_, \_, \_) \rightarrow (\_, \{\Gamma \vdash_\ell P\,, \Gamma \vdash_\ell Q\} \cup J_p\,, \_, \_, \_)$$

(p₄ₐ) $$(\_, \{\Gamma \vdash_\ell [p = q]P, Q\} \cup J_p\,, \_, C\,, \_) \rightarrow (\_, \{\Gamma \vdash_\ell P\,,\ \Gamma \vdash_\ell Q\} \cup J_p\,, \_, \\ C \cup \{\Gamma(a) \doteq \mathit{val}\,, \Gamma(b) \doteq \mathit{val}\}\,, \_)$$

(p₄ᵦ) $$(\_, \{\Gamma \vdash_\ell [\ell_1 = \ell_2]P, Q\} \cup J_p\,, \_, C\,, \_) \rightarrow (\_, \{\Gamma \vdash_\ell P\,,\ \Gamma \vdash_\ell Q\} \cup J_p\,, \_, \\ C \cup \{\Gamma(\ell_1) \doteq \rho_\emptyset\,, \Gamma(\ell_2) \doteq \rho'_\emptyset\}\,, \_)$$

(p₅) $$(\_, \{\Gamma \vdash_\ell (\nu a)P\} \cup J_p\,, \_, C\,, \_) \rightarrow (\_, \{\Delta \vdash_\ell P\} \cup J_p\,, \_, C \cup C'\,, \_)$$

$C' = \{\rho_L \doteq \rho'_{L\cup\{a\}}\}$ and $\Delta =_{\mathit{dom}(\Gamma)-\ell} \Gamma,\ \Delta(\ell) = [\{a : h, \rho'_{L\cup\{a\}}\}/\rho_L]\Gamma(\ell)$ where $\rho_L = \rho\mathit{var}(\Gamma(\ell))$

(p₆) $$(\_, \{\Gamma \vdash_k (\nu a@\ell)P\} \cup J_p\,, \_, C\,, \_) \rightarrow (\_, \{\Delta \vdash_k P\} \cup J_p\,, \_, C \cup C'\,, \_)$$

same side condition as for *(p₅)*

(p₇) $$(\_, \{\Gamma \vdash_\ell (\nu p)P\} \cup J_p\,, \_, \_, \_) \rightarrow (\_, \{p : \mathit{val}, \Gamma \vdash_\ell P\} \cup J_p\,, \_, \_, \_)$$

(p₈) $$(\_, \{\Gamma \vdash_\ell (\nu k)P\} \cup J_p\,, \_, \_, \_) \rightarrow (\_, \{k : \rho_\emptyset, \Gamma \vdash_\ell P\} \cup J_p\,, \_, \_, \_)$$

(p₉) $$(\_, \{\Gamma \vdash_\ell \mathsf{go}\ k.P\} \cup J_p\,, \_, \_, \_) \rightarrow (\_, \{\Gamma \vdash_k P\} \cup J_p\,, \_, \_, \_)$$

(p₁₀) $$(\_, \{\Gamma \vdash_\ell A(u)\} \cup J_p\,, J_n\,, C\,, \_) \rightarrow (\_, J_p\,, \{\Gamma \vdash^W_\ell u : t\} \cup J_n\,, \\ C \cup \{\Gamma(A) \doteq \mathit{Ch}(t)\}\,, \_)$$

(p₁₁) $$(\_, \{\Gamma \vdash_\ell T(u)\} \cup J_p\,, J_n\,, \_, \_) \rightarrow (\_, \{\Gamma \vdash_\ell T : \mathit{Ch}(t)\} \cup J_p\,, \\ \{\Gamma \vdash^W_\ell u : t\} \cup J_n\,, \_, \_)$$

if $T \notin \mathcal{P}$

(p₁₂) $$(\_, \{\Gamma \vdash_\ell \mathsf{rec}\ A(u).P : \mathit{Ch}(\tau)\} \cup J_p, \_, C, A) \rightarrow (\_, \{A : \mathit{Ch}(\tau), \Delta \vdash_\ell P\} \cup J_p\,, \_, \\ C \cup C'' \cup \{t \doteq \tau\}\,, A \cup A')$$

same side condition as for *(p₂)*

(pc) $$(\_, \{\Gamma \vdash_\ell P\} \cup J_p\,, \_, \_, \_) \rightarrow \bot$$

if Γ is not well formed or if none of the previous rules can be applied to this sequent.

Figure 8: Generation of constraints for processes

$$\begin{array}{lrcl}
\textit{(net}_0\textit{)} & (\{\Gamma \vdash \mathbf{0}\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & (J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) \\
\textit{(net}_1\textit{)} & (\{\Gamma \vdash [\ell :: P]\} \cup J_s\,,\,J_p\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & (J_s\,,\,\{\Gamma \vdash_\ell P\} \cup J_p\,,\,\_\,,\,\_\,,\,\_) \\
\textit{(net}_2\textit{)} & (\{\Gamma \vdash S \mid S'\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & (\{\Gamma \vdash S\,,\,\Gamma \vdash S'\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) \\
\textit{(net}_3\textit{)} & (\{\Gamma \vdash (\nu a@\ell)S\} \cup J_s\,,\,\_\,,\,\_\,,\,C\,,\,\_) & \rightarrow & (\{\Delta \vdash S\} \cup J_s\,,\,\_\,,\,\_\,,\,C \cup C'\,,\,\_) \\
\end{array}$$

$$C' = \{\rho_L \doteq \rho'_{L\cup\{a\}}\} \text{ and } \Delta =_{dom(\Gamma)-\ell} \Gamma,\ \Delta(\ell) = [\{a : h, \rho'_{L\cup\{a\}}\}/\rho_L]\Gamma(\ell),\ \rho_L = \rho var(\Gamma(\ell))$$

$$\begin{array}{lrcl}
\textit{(net}_4\textit{)} & (\{\Gamma \vdash (\nu \ell)S\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & (\{\ell : \rho_\emptyset, \Gamma \vdash S\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) \\
\textit{(net}_5\textit{)} & (\{\Gamma \vdash (\nu p)S\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & (\{p : \mathit{val}, \Gamma \vdash S\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) \\
\textit{(net\_clash)} & (\{\Gamma \vdash S\} \cup J_s\,,\,\_\,,\,\_\,,\,\_\,,\,\_) & \rightarrow & \bot
\end{array}$$

if Γ is not well formed or if none of the previous rules can be applied to this sequent.

Figure 9: Generation of constraints for systems

Let us comment the reduction rules are collected in figures 7 to 9. An underscore denotes a meaningless component. In figure 7, the rules concern sequents for names. We can remark that the rules *(n*$_1$*)* and *(n*$_3$*)* add a subtyping inequation to A. The rules *(n*$_2$*)* to *(n*$_4$*)* generate constraints between the typing of the name given by the context and the one expected. For instance, in the rule *(n*$_3$*)* we expect a type τ for the channel name a at remote location k. Then, in accordance with the corresponding rule of the type system, we constraint the type for k in Γ to type a with a variable h and τ to be unifiable with $h@\rho'_\emptyset$. Moreover, the type given for k in Γ have to assign types to at least the channels expected by τ for k. In these rules $h, \rho'_\emptyset$ and $\rho_{\{a\}}$ are supposed to be fresh that is they have no occurrence in the initial tuple.

Sequents in J_n are added when applying rules of figure 8. For instance, *(p*$_1$*)* generates, from a judgment for a message $\overline{a}u$, a sequent in J_n for typing the object u with a type variable t. A constraint (C') is also generated to specify that the context must provide at least a type for the current location that types a with $Ch(t)$. The rules *(p*$_{10}$*)* and *(p*$_{11}$*)* for typing the application of a recursive process are very similar.

In the presence of an input process, the type system uses the auxiliary strong type system for names. Since this one is very simple and completely deterministic, given a name u, a type τ and a location ℓ we can easily determine the context Γ such that $\Gamma \vdash_\ell u : \tau$. Actually, for our purpose we only need τ to be a type variable. This is done by the function $gen(u : t, \ell)$ that provides a context and a typing problem.

Definition 5.1 *We define the function* $gen(u : t, \ell) = (\Gamma, C, A)$ *as follows:*

$$\begin{array}{rcl}
gen(p : t, \ell) & = & (p : \mathit{val}, \{t \doteq \mathit{val}\}, \emptyset) \\
gen(k : t, \ell) & = & (k : \rho_\emptyset, \{t \doteq \rho_\emptyset\}, \emptyset) \\
gen(a@k : t, \ell) & = & (k : \{a : h, \rho_{\{a\}}\}, \{t \doteq h@\rho'_\emptyset\}, \{\rho'_\emptyset \doteqdot \rho_{\{a\}}\})
\end{array}$$

where $\ell \neq k$ *and* ρ, ρ' *and* h *are fresh.*

The typing problem associated with the context provided by this function is intended to force the type variable to have a valid form. For instance, to type $a@k$, the type variable must unify with a located type. Therefore, the idea is that if $gen(u : t, \ell) = (\Gamma, C, A)$, and (C, A) has a solution μ then $\mu\Gamma \vdash_\ell u : \mu t$ is expected to be valid in the strong type system for names.

In the rule *(p*$_2$*)* we make use of this function when the received name is not a simple one. In this case we type the body of the input in the initial context extended with the context provided by *gen*. This extension does not give rise to any problem since the name(s) received does not already occur in the context and all type variables of the extending

context are assumed to be fresh. For instance, if $u = a@k$ then, the extending context is $k : \{a : h, \rho_{\{a\}}\}$ and, since k was a bound name, Γ does not assign a type to k. Finally, the typing problem C is completed with the one of the function *gen* and a constraint on the typing of the current location assigned by Γ as in *(p_1)*.
When the name received is simple (say b), as in the type system, the type ψ assigned by Γ for the current location ℓ have to be extended with a type assignment for b. This extension is performed by the substitution of the row variable ρ_L of ψ for a location type assigning a type to b. The final context Δ by this way obtained is described in side condition of *(p_2)*. In order to keep the coherence between the new row variable of the type assigned by Δ to ℓ and the one previously assigned by Γ and possibly still occurring in the remaining tuple, we add a constraint equating these two row variables (C'' in side condition of *(p_2)*).

Example 5.2 As a small example, suppose we want to generate a problem for the network

$$S = [\ell :: \overline{a}b]$$

Starting with the minimal context $\ell : \rho_\emptyset$ the sequence of reductions is the following:

$$\begin{array}{lll} & (\{\ell : \rho_\emptyset \vdash [\ell :: \overline{a}b]\}, \emptyset, \emptyset, \emptyset, \emptyset) & \\ \twoheadrightarrow & (\emptyset, \{\ell : \rho_\emptyset \vdash_\ell \overline{a}b\}, \emptyset, \emptyset, \emptyset) & (net_1) \\ \twoheadrightarrow & (\emptyset, \emptyset, \{\ell : \rho_\emptyset \vdash^W_\ell b : t\}, \{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}\}, \emptyset) & (p_1) \\ \twoheadrightarrow & (\emptyset, \emptyset, \emptyset, \{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}, \rho_\emptyset \doteq \{b : t, \rho''_{\{b\}}\}\}, \emptyset) & (n_2) \end{array}$$

Thus, the problem generated $(\{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}, \rho_\emptyset \doteq \{b : t, \rho''_{\{b\}}\}\}, \emptyset)$ has the following most general solution w.r.t. $\{t, \rho_\emptyset, \rho'_{\{a\}}, \rho''_{\{b\}}\}$:

$$\mu = [\{a : Ch(t), b : t, \rho'''_{\{a,b\}}\}/\rho_\emptyset, \ \{b : t, \rho'''_{\{a,b\}}\}/\rho'_{\{a\}}, \ \{a : Ch(t), \rho'''_{\{a,b\}}\}/\rho''_{\{b\}}]$$

Applying μ to the initial context we obtain the valid sequent:

$$\ell : \{a : Ch(t), b : t, \rho'''_{\{a,b\}}\} \vdash [\ell :: \overline{a}b]$$

□

Now, we can turn to the main results concerning the reduction relation we have described. The first one shows the termination of the rewriting system.

Lemma 5.3 (Terminaison) *Given (J_s, J_p, J_n, C, A), a sequence of reductions*

$$(J_s, J_p, J_n, C, A) \twoheadrightarrow (J'_s, J'_p, J'_n, C', A') \twoheadrightarrow \ldots$$

terminates either on $\perp$ or on $(\emptyset, \emptyset, \emptyset, C'', A'')$.

The following lemma establishes the preservation of the solutions of the tuples along the reduction steps. By solution of a tuple (J_s, J_p, J_n, C, A) we mean a substitution that is solution of the problem (C, A) and that validates the sequents of J_s, J_p and J_n.

Lemma 5.4 (Preservation) *Let μ solution of (J_s, J_p, J_n, C, A) and λ of $(J'_s, J'_p, J'_n, C', A')$.*
If $(J_s, J_p, J_n, C, A) \twoheadrightarrow (J'_s, J'_p, J'_n, C', A')$ then

1. *there is $\lambda' =_{\mathcal{X}} \mu$ solution of $(J'_s, J'_p, J'_n, C', A')$, and $\mathcal{X} = var(J_s, J_p, J_n, C, A)$,*
2. *λ is also solution of (J_s, J_p, J_n, C, A).*

We informally spoke about initial contexts in the introduction of this section. In order to state the theorem of this paper, we give the formal definition.

Definition 5.5 *An* initial context *for a system S is a context*

$$\Gamma = \{\ell_1 : \rho^1_\emptyset, \ldots, \ell_n : \rho^n_\emptyset, p_1 : val, \ldots, p_m : val\}$$

such that location and value names and row variables are distincts.

Theorem 5.6 (Soundness) *Let S be a network and Γ be an initial context for S, if*

$$(\{\Gamma \vdash S\}\,,\,\emptyset\,,\,\emptyset\,,\,\emptyset\,,\,\emptyset) \twoheadrightarrow^* (\emptyset\,,\,\emptyset\,,\,\emptyset\,,\,C\,,\,A) \text{ and } (C\,,\,A\,,\,\emptyset) \rightarrow^* (\emptyset\,,\,A'\,,\,\mu)$$

then $\mu\Gamma; A'$ is a principal typing for S.

Finally, we state the completeness theorem for networks.

Theorem 5.7 (Completeness) *Let S be a typable network and Γ be an initial context for S' then*

$$(\{\Gamma \vdash S\}\,,\,\emptyset\,,\,\emptyset\,,\,\emptyset\,,\,\emptyset) \twoheadrightarrow^* (\emptyset\,,\,\emptyset\,,\,\emptyset\,,\,C\,,\,A) \text{ and } (C\,,\,A\,,\,\emptyset) \rightarrow^* (\emptyset\,,\,A'\,,\,\mu)$$

6 Conclusion

In this paper we studied the problem of type inference for the type system of $\mathrm{D}\pi_1^r$ ([ABL00]) a distributed π-calculus with code migration and local communication. We first motivated some modifications of the original type system. We also extended it with existential types $\gamma@\psi$ in order to be able to type more systems and to be closer to the type system of [HR98]. The use of an explicit subtyping relation on location types involving location type variables allowed us to define a notion of principal typing. The latter being a typing context assigning type schemes to names together with a set of atomic subtyping assertions. Next, we addressed the problem of unification of type schemes. As we were interested in solving equations between types, we were also interested in solving inequations between location types. We then defined an algorithm, by means of a rewriting relation, that, given a set of equations and inequations (that we called a problem), provides their solution, that is a most general substitution. We saw that unification of location types is very close to unification of record types (see for instance [Rém93a, Rém93b, JM93]). Finally, we gave a algorithm that, given a system S together with its initial context, computes a problem whose solution applied to the initial context is a principal typing for S. We stated the soundness and completeness properties of this algorithm.

For the sake of simplicity, in this paper we considered the type system for a monadic version of the calculus. However, an implementation of the algorithms have been made for the full polyadic calculus.

We believe that this work should be easily adapted to the type system of [HR98]. Moreover, the presentation of algorithms we adopted, that is by means of reduction relations, and the fact that we compute a principal typing, should be useful for a formal definition of dynamic typing (and thus for type checking) and its integration in process reduction. For instance, Hennessy and Reily in [RH98], study partial typing for open systems of mobile agents where only some sites may be typed. When an agent intends to move from a location k into a well typed location ℓ, its code is dynamically type checked with respect to the typing context embedded for ℓ (called a *filter*). The latter represents partial knowledge of ℓ of the rest of the world and in particular of the location k. However, on one hand, they informally assume the existence of a type checker and, on the other hand, terms involve typing information. We think that their work could be extended to allow dynamic computation of type information.

References

[ABL00] R. Amadio, G. Boudol, and C. Lhoussaine. The receptive distributive π-calculus. Technical Report 4080, INRIA, Sophia Antipolis, 2000.

[Ama97] R. M. Amadio. An asynchronous model of locality, failure, and process mobility. In D. Garlan and D. Le Métayer, editors, *Proceedings of the 2nd International Conference on Coordination Languages and Models (COORDINATION'97)*, volume 1282, pages 374–391, Berlin, Germany, 1997. Springer-Verlag.

[BB90] G. Berry and G. Boudol. The Chemical Abstract Machine. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 81–94, San Francisco, California, January 17–19, 1990. ACM Press, New York.

[BS99] F. Baader and W. Snyder. *Handbook of Automated Reasoning*, chapter Unification theory. Elsevier Science Publishers, 1999. To appear.

[CG98] L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 1378, pages 140–155. Springer-Verlag, Berlin, Germany, 1998.

[FGL$^+$96] C. Fournet, G. Gonthier, J-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96)*, pages 406–421. Springer-Verlag, 1996.

[HR98] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. Technical Report 2/98, School of Cognitive and Computer Sciences, University of Sussex, 1998.

[JM93] L. Jategaonkar and J. C. Mitchell. Type inference with extended pattern matching and subtypes. *Fundamenta Informaticae*, 19(1/2):127–165, 1993.

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh, 1991.

[PS93] B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes, 1993.

[Rém93a] D. Rémy. Syntactic theories and the algebra of record terms. Research Report 1869, Institut National de Recherche en Informatique et Automatisme, Rocquencourt, BP 105, 78 153 Le Chesnay Cedex, France, 1993.

[Rém93b] D. Rémy. Type inference for records in a natural extension of ML. In Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects Of Object-Oriented Programming. Types, Semantics and Language Design.* MIT Press, 1993.

[RH98] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. Technical Report 4, University of Sussex, 1998.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:227–234, 1965.

[Tur95] D. Turner. *The Polymorphic Pi-Calculus: Theory and Implementation.* PhD thesis, University of Edinburgh, 1995.

[Wan87] M. Wand. Complete type inference for simple objects. In *Symposium on Logic in Computer Science, Ithaca, NY*, pages 37–44, 1987.

A General Definitions

In this section we give some formal definitions needed for the proofs given in the next sections. We define

$$nm(u) = \begin{cases} \{a, \ell\} & \text{if } u = a@\ell \\ \{u\} & \text{otherwise} \end{cases} \qquad subj(u) = \begin{cases} \{a\} & \text{if } u = a@\ell \\ \{u\} & \text{otherwise} \end{cases}$$

We use $x, y, \ldots$ to denote simple names of any kind.

Definition A.1 *We note $fn(P)$ the set of free names occuring in the (possibly parametric) process P and defined as follows:*

$$\begin{array}{rcl} fn(\mathbf{0}) & = & \emptyset \\ fn(\overline{a}u) & = & \{a\} \cup nm(u) \\ fn(a(u).P) & = & (\{a\} \cup fn(P)) - nm(u) \\ fn(P \mid Q) & = & fn(P) \cup fn(Q) \\ fn([x = y]P, Q) & = & \{x, y\} \cup fn(P) \cup fn(Q) \\ fn((\nu u)P) & = & fn(P) - subj(u) \\ fn(\mathsf{go}\, \ell.P) & = & \{\ell\} \cup fn(P) \\ fn(T(u)) & = & nm(u) \\ fn(\mathsf{rec}\, A(u).P) & = & fn(P) - nm(u) \end{array}$$

We also note $fn(S)$ the set of free names occuring in the network S and defined as follows:

$$\begin{array}{rcl} fn(\mathbf{0}) & = & \emptyset \\ fn([\ell :: P]) & = & \{\ell\} \cup fn(P) \\ fn(S \mid S') & = & fn(S) \cup fn(S') \\ fn((\nu a@\ell)S) & = & fn(S) - \{a\} \\ fn((\nu p)S) & = & fn(S) - \{p\} \\ fn((\nu \ell)S) & = & fn(S) - \{\ell\} \end{array}$$

Definition A.2 *We note $bn(P)$ the set of bound names occuring in the process P and defined as follows:*

$$bn(\overline{a}u) = bn(\mathbf{0}) = bn(A) = \emptyset$$

$$bn(a(u).P) = bn(\mathsf{rec}\, A(u).P) = nm(u) \cup bn(P)$$

$$bn((\nu u)P) = subj(u) \cup bn(P)$$

$$bn(P \mid Q) = bn([x = y]P, Q) = bn(\mathsf{go}\, \ell.P) = bn(P)$$

$$bn(T(u)) = bn(T)$$

We also note $bn(S)$ the set of bound names occuring in S defined as follows:

$$\begin{array}{rcl} bn(\mathbf{0}) & = & \emptyset \\ bn((\nu a@\ell)S) & = & \{a\} \cup bn(S) \\ bn((\nu p)S) & = & \{p\} \cup bn(S) \\ bn((\nu \ell)S) & = & \{\ell\} \cup bn(S) \\ bn([\ell :: P]) & = & bn(P) \\ bn(S \mid S') & = & bn(S) \cup bn(S') \end{array}$$

B Subject Reduction

B.1 Operational Semantics

We define the *structural equivalence* as the least equivalence[4] $\equiv$ containing the following axioms:

$$\begin{array}{rcll}
(\mathbf{0} \mid U) & \equiv & U & \\
(U \mid (V \mid W)) & \equiv & ((U \mid V) \mid W) & \\
(U \mid V) & \equiv & (V \mid U) & \\
((\nu u)U \mid V) & \equiv & (\nu u)(U \mid V) & \mathit{subj}(u) \notin \mathit{fn}(V) \\
[\ell :: P \mid Q] & \equiv & [\ell :: P] \mid [\ell :: Q] & \\
[\ell :: (\nu u)P] & \equiv & (\nu u@\ell)[\ell :: P] & \ell \notin \mathit{subj}(u)
\end{array}$$

where

$$u@\ell = \begin{cases} a@\ell & \text{if } u = a \\ u & \text{otherwise} \end{cases}$$

and satisfying the following rules

$$\frac{U =_\alpha V}{U \equiv V} \qquad \frac{U \equiv V \;,\; V \equiv W}{U \equiv W} \qquad \frac{U \equiv V}{\mathbf{E}[U] \equiv \mathbf{E}[V]}$$

where $\mathbf{E}$ is any *evaluation context*, given by the following grammar:

$$\mathbf{E} ::= \square \mid (\mathbf{E} \mid U) \mid (\nu w)\mathbf{E} \mid [\ell :: \mathbf{E}]$$

Reduction is defined up to structural equivalence and may occur in any evaluation context:

$$\frac{V \equiv U \;,\; U \to U' \;,\; U' \equiv V'}{V \to V'} \qquad \frac{U \to U'}{\mathbf{E}[U] \to \mathbf{E}[U']}$$

B.2 Substitutions

A name substitution is a mapping S from a finite subset $\mathit{dom}(\mathsf{S})$ of $\mathcal{N}$ into $\mathcal{N}$, such that $\forall x \in \mathcal{N}_{chan}$ (resp. $\mathcal{N}_{val}, \mathcal{N}_{loc}$), $\mathsf{S}(x) \in \mathcal{N}_{chan}$ (resp. $\mathcal{N}_{val}, \mathcal{N}_{loc}$). We denote by $[\mathsf{S}]U$ the application of S to U if $\mathit{im}(\mathsf{S}) \cap \mathit{bn}(U) = \emptyset$. We give the definition of the substitution for the binding constructs, where by convention $\mathsf{S}(x) = x$ if $x \notin \mathit{dom}(\mathsf{S})$:

$$\begin{array}{rcll}
[\mathsf{S}]a(u).P & = & \mathsf{S}(a)(u).[\mathsf{S}']P & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \mathit{nm}(u)) \\
[\mathsf{S}](\nu a)P & = & (\nu a)[\mathsf{S}']P & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{a\}) \\
[\mathsf{S}](\mathsf{rec}\, A(u).P) & = & (\mathsf{rec}\, A(u).[\mathsf{S}']P) & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \mathit{nm}(u)) \\
[\mathsf{S}](\nu \ell)U & = & (\nu \ell)[\mathsf{S}']U & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{\ell\}) \\
[\mathsf{S}](\nu p)U & = & (\nu p)[\mathsf{S}']U & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{p\}) \\
[\mathsf{S}](\nu a@\ell)U & = & (\nu a@\mathsf{S}(\ell))[\mathsf{S}']U & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{a\})
\end{array}$$

The substitutions also apply to row variables so that $\mathsf{S}(\rho_L) = \rho_{\mathsf{S}(L)}$. Moreover, we assume that a substitution preserves well-formedness, that is if $\psi :: L$ is valid then so is $\mathsf{S}(\psi) :: \mathsf{S}(L)$. To this aim, substitutions applied to location types may involve implicit contractions as described by the following:

$$[{}^{a}/_{b}]\{a : \gamma, b : \gamma', \psi\} \quad = \quad \{a : \gamma'', [{}^{a}/_{b}]\psi\}$$

if $\gamma'' = [{}^{a}/_{b}]\gamma = [{}^{a}/_{b}]\gamma'$. Substitutions on typing contexts may also involve implicit contractions:

$$\begin{array}{rcll}
[{}^{\ell}/_{k}](\ell : \psi, k : \psi', \Gamma) & = & \ell : \psi, \Gamma & \text{if } \psi \sqsubseteq \psi' \\
[{}^{p}/_{q}](p : \mathit{val}, q : \mathit{val}, \Gamma) & = & p : \mathit{val}, \Gamma &
\end{array}$$

The relation $=_\alpha$ of α-conversion is the least congruence satisfying the following axioms:

[4]there should be no confusion with the equivalence on location types.

$$\begin{array}{rcll}
a(u).P & =_\alpha & a(v).[v/u]P & nm(u) \cap (nm(P) \cup \{a\}) = \emptyset \\
((\nu x)U) & =_\alpha & (\nu y)[y/x]U & y \notin nm(U) \\
((\nu a@\ell)U) & =_\alpha & (\nu b@\ell)[b/a]U & b \notin nm(U) \cup \{\ell\} \\
(\mathsf{rec}\, A(u).P) & =_\alpha & (\mathsf{rec}\, B(v).[B/A][v/u]P) & B \notin nm(P), nm(v) \cap nm(P) = \emptyset
\end{array}$$

Lemma B.1 (Substitution) *The following holds:*

1. *if $\Gamma \vdash_\ell u : \tau$ (resp. $\Gamma \vdash_\ell^W u : \tau$) is provable, then so is $\mathsf{S}(\Gamma \vdash_\ell u : \tau)$ (resp. $\mathsf{S}(\Gamma \vdash_\ell^W u : \tau)$) for any substitution S such that $\mathsf{S}(\Gamma)$ is a typing context and S is injective on $nm(\tau)$;*

2. *if $\Gamma \vdash_\ell P$ (resp. $\Gamma \vdash_\ell T : Ch(\tau)$) is provable with a proof of heigth h, then so is $\mathsf{S}(\Gamma \vdash_\ell P)$ (resp. $\mathsf{S}(\Gamma \vdash_\ell T : Ch(\tau))$) for any substitution S such that $[\mathsf{S}]P$ (resp. $[\mathsf{S}]T$) is defined and $\mathsf{S}(\Gamma)$ is a typing context (and S is injective on $nm(\tau)$);*

3. *if $\Gamma \vdash S$ is provable with a proof of heigth h, then so is $\mathsf{S}(\Gamma \vdash S)$, for any substitution S such that S is injective on $nm(\mathsf{S})$, $[\mathsf{S}]S$ is defined and $\mathsf{S}(\Gamma)$ is a typing context;*

4. *if $\Gamma \vdash_\ell T : Ch(\tau)$ and $A : Ch(\tau), \Gamma \vdash_\ell P$ (resp. $A : Ch(\tau), \Gamma \vdash_\ell T' : \sigma$) are provable, and A does not occur in Γ, then $\Gamma \vdash_\ell [T/A]P$ (resp. $\Gamma \vdash_\ell [T/A]T' : \sigma$) is provable, provided that no free name of T is bound in P (resp. T').*

Proof Sketch: (1) It relies on the fact that $\sqsubseteq$ is preserved by substitution.
(2) By induction on h, and by case on the last rule used to infer the sequent. If P is $a(b).R$ and $\Gamma \vdash_\ell a(b).R$ and $\Delta \vdash_\ell R$ where

$$\Delta =_{dom(\Gamma)-\ell} \Gamma \text{ and } \Delta(\ell) = [\{b : \gamma, \rho_{L\cup\{b\}}\}/\rho_L]\Gamma(\ell)$$

and $\Gamma(\ell) \sqsubseteq \{a : Ch(\gamma)\}$, then $\mathsf{S}'(\Delta) \vdash_{\ell'} [\mathsf{S}']R$ is provable by induction hypothesis, where $\mathsf{S}' = \mathsf{S} \upharpoonright dom(\mathsf{S}) - \{b\}$ and $\ell' = \mathsf{S}'(\ell)$. We observe that substitutions preserve $\sqsubseteq$ and

$$\mathsf{S}'(\Delta) =_{dom(\mathsf{S}'(\Gamma))-\ell'} \mathsf{S}(\Gamma) \text{ and } \mathsf{S}'(\Delta)(\ell') = [\{b : \mathsf{S}'(\gamma), \rho_{\mathsf{S}'(L)\cup\{b\}}\}/\rho_{\mathsf{S}'(L)}]\mathsf{S}'(\Gamma)(\ell')$$

because $b \notin dom(\mathsf{S}')$. Then, since $b \notin im(\mathsf{S})$ we may infer $\mathsf{S}(\Gamma \vdash_\ell a(b).P)$.
If $P = a(u).R$ where u is not a channel name, $\Gamma \vdash_\ell P$ and $\Gamma, \Delta \vdash_\ell R$ where $\Gamma \sqsubseteq \{a : Ch(\tau)\}$ and $\Delta \vdash_\ell u : \tau$, then $\mathsf{S}'(\Gamma, \Delta \vdash_\ell R)$ is provable, by induction hypothesis, where $\mathsf{S}' = \mathsf{S} \upharpoonright (dom(S) - nm(u))$. Since $\mathsf{S}'(\Delta) \vdash_\ell u : \mathsf{S}'(\tau)$, we may the rule for typing the input construct to conclude. □

Lemma B.2 *Let S be an injective substitution such that $\Gamma \vdash_\ell [\mathsf{S}]P$ (resp. $\Gamma \vdash [\mathsf{S}]S$) is provable, with a proof of height h. Then so is $\mathsf{S}^{-1}(\Gamma) \vdash_{\mathsf{S}^{-1}(\ell)} P$ (resp. $\mathsf{S}(\Gamma) \vdash S$).*

Proof Sketch: By induction on the inference of sequent, and by case on the last rule used this inference. We examine the case where $[\mathsf{S}]P = a(b).R$, $\Gamma \vdash_\ell a(b).R$ and $\Delta \vdash_\ell R$. We have $P = a'(b).R'$ with $a' = \mathsf{S}(a)$ and $R = [\mathsf{S}_0]R'$ where $\mathsf{S}_0 = \mathsf{S} \upharpoonright dom(\mathsf{S}) - \{b\}$. Since we are assuming that $[\mathsf{S}]P$ is defined, we have $b \notin im(\mathsf{S})$. By induction hypothesis $\mathsf{S}_0^{-1}(\Delta) \vdash_{\ell'} R'$ where $\ell' = \mathsf{S}^{-1}(\ell)$. Since b does not occur in Γ and $\Gamma \sqsubseteq \{a : Ch(\tau)\}$, we have $[\mathsf{S}_0^{-1}]\tau = [\mathsf{S}^{-1}]\tau$ and $\mathsf{S}_0^{-1}(\Gamma) = \mathsf{S}^{-1}(\Gamma)$. Let $\theta = [\mathsf{S}^{-1}]\tau$. Then, we can observe that

$$\mathsf{S}_0^{-1}(\Delta) =_{dom(\mathsf{S}^{-1}(\Gamma))-\ell'} \mathsf{S}^{-1}(\Gamma)$$

and

$$\mathsf{S}_0^{-1}(\Delta)(\ell') = [\{b : \theta, \rho'_{L'\cup\{b\}}\}/\rho_{L'}]\mathsf{S}^{-1}(\Gamma)(\ell')$$

where $L' = \mathsf{S}^{-1}(L)$. Moreover, $S^{-1}(\Gamma)(\ell') \sqsubseteq \{a' : Ch(\theta)\}$, therefore we can infer $\mathsf{S}^{-1}(\Gamma) \vdash_{\ell'} P$. □

Lemma B.3 *If $P =_\alpha Q$ (resp. $S =_\alpha S'$) and $\Gamma \vdash_\ell P$ (resp. $\Gamma \vdash S$) then $\Gamma \vdash_\ell Q$ (resp. $\Gamma \vdash S'$).*

Proof Sketch: By induction on the definition of $=_\alpha$. Since, the rules of the type system only depend on the form of the terms being typed, we just deal with the axioms of α-conversion. We show that if $\Gamma \vdash_\ell a(u).P$ and $[v/u]$ is a substitution such that $nm(v) \cap (nm(P) \cup \{a\}) = \emptyset$, then $\Gamma \vdash_\ell a(v).[v/u]P$. Let S be an injective substitution such that $dom(\mathsf{S}) = nm(v)$ and the names in $im(\mathsf{S})$ are fresh. Then by the lemma B.1 (2), $\mathsf{S}(\Gamma) \vdash_{\ell'} a(u).P$ where $\ell' = \mathsf{S}(\ell)$, is provable. We first consider the case where $u = b$, $v = c$ and $\mathsf{S} = [{}^{d}\!/_{c}]$ considering d as a fresh channel name (note that in this case $\ell' = \ell$). By the input rule we have $\mathsf{S}(\Gamma)(\ell) \sqsubseteq \{a : \gamma\}$ and $\Delta \vdash_\ell P$, where

$$\Delta =_{dom(\mathsf{S}(\Gamma)-\ell)} \mathsf{S}(\Gamma) \text{ and } \Delta(\ell) = [\{b : \gamma, \rho'_{L\cup\{b\}}\}/\rho_L]\mathsf{S}(\Gamma)(\ell)$$

and by the lemma B.1 (2), $[{}^{c}\!/_{b}]\Delta \vdash_\ell [{}^{c}\!/_{b}]P$ is provable. Moreover, since $b \notin L$ and does not occurs in $\mathsf{S}(\Gamma)$, we have

$$[{}^{c}\!/_{b}]\Delta =_{dom(\mathsf{S}(\Gamma)-\ell)} \mathsf{S}(\Gamma) \text{ and } [{}^{c}\!/_{b}]\Delta(\ell) = [\{c : \gamma, \rho'_{L\cup\{c\}}\}/\rho_L]\mathsf{S}(\Gamma)(\ell)$$

Since we have chosen S such that c does not occur in $\mathsf{S}(\Gamma)$, we may infer $\mathsf{S}(\Gamma) \vdash_\ell a(c).[{}^{c}\!/_{b}]P$. Therefore, $\Gamma \vdash_\ell a(c).[{}^{c}\!/_{b}]P$ by the lemma B.2, since $a(c).[{}^{c}\!/_{b}]P = [\mathsf{S}]a(c).[{}^{c}\!/_{b}]P$.
In the case where u is not a simple name, we have $\mathsf{S}(\Gamma), \Delta \vdash_{\ell'} P$ where $\Delta \vdash_\ell u : \tau$ and $\mathsf{S}(\Gamma) \sqsubseteq \{a : Ch(\tau)\}$ and the names in $nm(u)$ do not occur in $\mathsf{S}(\Gamma)$. By the lemma B.1 (1), we have $[{}^{v}\!/_{u}]\Delta \vdash_{\ell'} v : \tau$ (the names of u cannot occur in τ, since they do not occur in $\mathsf{S}(\Gamma)$), and by the lemma B.1 (2) we may infer $\mathsf{S}(\Gamma), [{}^{v}\!/_{u}]\Delta \vdash_{\ell'} [{}^{v}\!/_{u}]P$. And we conclude has in the previous case.
To prove the symmetric case, that is $\Gamma \vdash_\ell a(v).[{}^{u}\!/_{v}]P$ implies $\Gamma \vdash_\ell a(u).P$, we proceed in a similar way, first renamming the names of u (which do not include a) in Γ with fresh names, and using the lemma B.2. The cases for restrictions are similar. □

Definition B.4 $\Gamma \sqsubseteq \Delta$ *iff* $dom(\Delta) \subseteq dom(\Gamma)$ *and* $\forall \ell \in dom(\Delta).\, \Gamma(\ell) \sqsubseteq \Delta(\ell)$.

Lemma B.5 *If Γ and Δ are typing contexts such that*

$$\Delta =_{dom(\Gamma)-\ell} \Gamma \;\; \textit{and} \;\; \Delta(\ell) = [\{a : \gamma, \rho'_{L\cup\{a\}}\}/\rho_L]\Gamma(\ell)$$

where $\rho_L = \rho var(\Gamma(\ell))$, then $\Delta \sqsubseteq \Gamma$.

Lemma B.6 (Weakening)

1. *If $\Gamma \vdash_\ell P$, $\Delta \sqsubseteq \Gamma$ and $bn(P) \cap nm(\Delta) = \emptyset$ then $\Delta \vdash_\ell P$.*
2. *If $\Gamma \vdash_\ell T : \tau$, $\Delta \sqsubseteq \Gamma$ and $bn(T) \cap nm(\Delta) = \emptyset$ then $\Delta \vdash_\ell T : \tau$.*
3. *If $\Gamma \vdash S$, $\Delta \sqsubseteq \Gamma$ and $bn(S) \cap nm(\Delta) = \emptyset$ then $\Delta \vdash S$.*

Proof Hint: We proceed by induction on the inference of the sequent. We also have to show that if $\Delta \vdash^W_\ell u : \tau$ and $\Gamma \sqsubseteq \Delta$ then $\Delta \vdash^W_\ell u : \tau$. □

When we write $x \notin (\Gamma \vdash_\ell P)$ we mean that x does not occur in Γ (either in the domain or in the types assigned by the context), nor in P and that $x \neq \ell$.

Lemma B.7 (Strengthening) *Given Γ with $\Gamma(k) = \{a : \gamma, \psi\}$ and Δ such that $\Delta =_{dom(\Gamma)-k} \Gamma$ and $\Delta(k) = [\rho'_L/\rho_{L\cup\{a\}}]\psi$ with $\rho_{L\cup\{a\}} = \rho var(\psi)$, ρ'_L fresh and $a \notin L$; the following holds*

1. *If $\Gamma \vdash_\ell P$ is provable and $a \notin (\Delta \vdash_\ell P)$ then $\Delta \vdash_\ell P$ is provable.*
2. *If $\Gamma, \ell' : \psi' \vdash_\ell P$ is provable and $\ell' \notin fn(P) \cup \{\ell\}$ then $\Gamma \vdash_\ell P$ is provable.*
3. *If $\Gamma, p : val \vdash_\ell P$ is provable and $p \notin fn(P)$ then $\Gamma \vdash_\ell P$ is provable.*
4. *If $\Gamma \vdash_\ell T : \tau$ is provable and $a \notin (\Delta \vdash_\ell T : \tau)$ then $\Delta \vdash_\ell T : \tau$ is provable.*
5. *If $\Gamma, \ell' : \psi' \vdash_\ell T : \tau$ is provable and $\ell' \notin fn(T) \cup \{\ell\}$ then $\Gamma \vdash_\ell T : \tau$ is provable.*
6. *If $\Gamma, p : val \vdash_\ell T : \tau$ is provable and $p \notin fn(T)$ then $\Gamma \vdash_\ell T : \tau$ is provable.*
7. *If $\Gamma \vdash S$ is provable and $a \notin (\Delta \vdash S)$ then $\Delta \vdash S$ is provable.*
8. *If $\Gamma, \ell' : \psi' \vdash S$ is provable and $\ell' \notin fn(S)$ then $\Gamma \vdash S$ is provable.*
9. *If $\Gamma, p : val \vdash S$ is provable and $p \notin fn(S)$ then $\Gamma \vdash S$ is provable.*

Proof Sketch: (1) We proceed by induction on the inference of the sequent. The cases of typing rules for $\mathsf{go}\,\ell.P$, parallel composition and replication are straightforward. We first remark that if $a \notin (\Gamma \vdash_\ell^W u : \tau)$ then $\Delta \vdash_\ell^W u : \tau$.

(out) Suppose that $\Gamma \vdash \bar{b}u$, then $\Gamma \sqsubseteq \{b : Ch(\tau)\}$ and $\Gamma \vdash_\ell u : \tau$. Since $a \notin (\Gamma \vdash_\ell \bar{b}u)$ and $\Gamma \sqsubseteq \{b : Ch(\tau)\}$ we have $a \notin (\Gamma \vdash_\ell u : \tau)$, therefore $\Delta \vdash_\ell u : \tau$. Also, $a \neq b$, then $\Delta \sqsubseteq \{b : Ch(\tau)\}$, and finally $\Delta \vdash_\ell \bar{a}u$.

(in) Suppose that $\Gamma \vdash_\ell b(c).P$ and $c \in fn(P)$, then $\Gamma' \vdash_\ell P$ and $\Gamma \sqsubseteq \{b : Ch(\gamma)\}$ where

$$\Gamma' =_{dom(\Gamma)-\ell} \Gamma \text{ and } \Gamma'(\ell) = [\{c : \gamma, \rho'_{L\cup\{c\}}\}/\rho_L]\Gamma$$

with $\rho_L = \rho var(\Gamma(\ell))$. Since $a \neq b$ we still have $\Delta(\ell) \sqsubseteq \{b : Ch(\gamma)\}$. Since $a \neq b$, by induction hypothesis, we have $\Delta' \vdash_\ell P$ where

$$\Delta' =_{dom(\Gamma')-k} \Gamma' \text{ and } \Delta'(k) = [\rho'''_{L'}/\rho''_{L'\cup\{a\}}]\Gamma'(k)$$

where $\rho''_{L'\cup\{a\}} = \rho var(\Gamma'(k))$. It remains to show that we can infer $\Delta'' \vdash_\ell b(c).P$ where Δ'' is equal to Δ up to renaming of row variables depending on whether $k = \ell$ or $k \neq \ell$.
Now suppose that $\Gamma \vdash_\ell b(c@k').P$, then $\Gamma, k' : \{c : \gamma, \psi'\} \vdash_\ell P$ and $\Gamma \sqsubseteq \{b : Ch(\gamma@\psi)\}$ where $\psi' = \psi$ and having chosen $\rho var(\psi') \neq \rho'_L$. We conclude directly, applying the induction hypothesis. The case of input of a location name is similar.

(ν) The cases for restrictions are similar to the ones of input.

(2) We remark that if $\Gamma, k : \psi \vdash_\ell^W u : \tau$ and $k \notin nm(u) \cup \{\ell\}$, then $\Gamma \vdash_\ell u : \tau$. We proceed by induction on the inference of the sequent. □

We define the relation $\preccurlyeq_{\mathcal{T}}$ as follows :

$$\begin{array}{rcl} P \preccurlyeq_{\mathcal{T}} Q & \Leftrightarrow & \forall\Gamma\forall\ell(\Gamma \vdash_\ell P \Rightarrow \Gamma \vdash_\ell Q) \\ T \preccurlyeq_{\mathcal{T}} T' & \Leftrightarrow & \forall\Gamma\forall\ell\forall\tau(\Gamma \vdash_\ell T : \tau \Rightarrow \Gamma \vdash_\ell T' : \tau) \\ S \preccurlyeq_{\mathcal{T}} S' & \Leftrightarrow & \forall\Gamma(\Gamma \vdash S \Rightarrow \Gamma \vdash S') \end{array}$$

We denote by $=_{\mathcal{T}}$ the associated equivalence relation.

Lemma B.8 *The relation $\preccurlyeq_{\mathcal{T}}$ contains the structural equivalence relation $\equiv$.*

Proof Sketch: We already proved that $=_{\mathcal{T}}$ contains the α-equivalence (lemma B.3). Moreover it is easy to see that $=_{\mathcal{T}}$ is a congruence. So, it is enough to show that for each axiom $U \equiv V$ we have $U \preccurlyeq_{\mathcal{T}} V$ and $V \preccurlyeq_{\mathcal{T}} U$. We just consider two axioms, the others are trivials. Let $\Gamma \vdash_\ell ((\nu u)P \mid Q)$ with $u = a \in \mathcal{N}_{chan}$ and $a \notin fn(Q)$. We may assume that a does not occur in Γ. The proof of this sequent has the following form:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash_\ell P}}{\Gamma \vdash_\ell (\nu a)P} \qquad \dfrac{\vdots}{\Gamma \vdash_\ell Q}}{\Gamma \vdash_\ell ((\nu a)P \mid Q)}$$

where $\Delta =_{dom(\Gamma)-\ell} \Gamma$ and $\Delta(\ell) = [\{a : \gamma, \rho'_{L\cup\{a\}}\}/\rho_L]\Gamma(\ell)$ where $\rho_L = \rho var(\Gamma(\ell))$. Since $a \notin (\Gamma \vdash_\ell Q)$, by the lemma B.6 (1) we also have $\Delta \vdash_\ell Q$, therefore $\Gamma \vdash_\ell (\nu a)(P \mid Q)$. The cases where $u \notin \mathcal{N}_{chan}$ are similar. Conversely, a proof of $\Gamma \vdash_\ell (\nu a)(P \mid Q)$ has the following form:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash_\ell P} \qquad \dfrac{\vdots}{\Delta \vdash_\ell Q}}{\Delta \vdash_\ell P \mid Q}}{\Gamma \vdash_\ell (\nu a)(P \mid Q)}$$

where Δ is defined as previously. Since $a \notin (\Gamma \vdash_\ell Q)$, by the lemma B.7 (1) we deduce that $\Gamma \vdash_\ell Q$ is provable, therefore $\Gamma \vdash_\ell ((\nu a)P \mid Q)$ is provable. The other cases are similar.

For $[\ell :: (\nu u)P] \equiv (\nu u@\ell)[\ell :: P]$ (and $u = a$), let us assume that $\Gamma \vdash [\ell :: (\nu a)P]$. The proof of this sequent has the following structure:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash_\ell P}}{\Gamma \vdash_\ell (\nu a)P}}{\Gamma \vdash [\ell :: (\nu a)P]}$$

We can then infer the following:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash_\ell P}}{\Delta \vdash [\ell :: P]}}{\Gamma \vdash (\nu a@\ell)[\ell :: P]}$$

For the symmetric case, and the other restrictions, the proofs are similar. □

Theorem B.9 (Subject Reduction) $U \to V \quad \Rightarrow \quad U \preccurlyeq_{\mathcal{T}} V$

Proof: by induction on the definition of $U \to V$. Since $\preccurlyeq_{\mathcal{T}}$ is a precongruence containing the structural equivalence relation, it is enough to consider the axioms of reduction. The cases $[k :: \mathsf{go}\,\ell.P] \to [\ell :: P]$, $[x = x]P, Q \to P$ and $[x = y]P, Q \to Q$ are trivial.

For the case of communication law $(\overline{a}c \mid a(b).P) \to [c/b]P$, let Γ be a context such that $\Gamma \vdash_\ell (\overline{a}c \mid a(b).P)$. Then the proof of this sequent has the following structure:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash^W_\ell c : \gamma}}{\Gamma \vdash_\ell \overline{a}c} \qquad \dfrac{\dfrac{\vdots}{\Delta \vdash_\ell P}}{\Gamma \vdash_\ell a(b).P}}{\Gamma \vdash_\ell (\overline{a}c \mid a(b).P)}$$

with $\Gamma(\ell) \sqsubseteq \{a : Ch(\gamma)\}$ and where $\Delta =_{dom(\Gamma)-\ell} \Gamma$ and $\Delta(\ell) = [\{b : \gamma, \rho'_{L\cup\{b\}}\}/\rho_L]\Gamma(\ell)$ and b does not occurs in Γ, and therefore not in γ. From $\Gamma \vdash^W_\ell c : \gamma$, it is easy to see that $[c/b]\Delta = [\rho'_L/\rho_L]\Gamma$. Moreover, by the lemma B.1(2), we have $[c/b]\Delta \vdash_\ell [c/b]P$ therefore $[\rho'_L/\rho_L]\Gamma \vdash_\ell [c/b]P$ and $\Gamma \vdash_\ell [c/b]P$.

Let us now consider the case where a location name is communicated $(\overline{a}k \mid a(k').P) \to [k/k']P$ and let Γ be a context such that $\Gamma \vdash_\ell (\overline{a}k \mid a(k').P)$. Then, the proof of this sequent has the following form:

$$\dfrac{\dfrac{\dfrac{\Gamma(k) \sqsubseteq \psi'}{\Gamma \vdash^W_\ell k : \psi'}}{\Gamma \vdash_\ell \overline{a}k} \qquad \dfrac{\dfrac{\vdots}{\Gamma, k' : \psi' \vdash_\ell P}}{\Gamma \vdash_\ell a(k').P}}{\Gamma \vdash_\ell (\overline{a}k \mid a(k').P)}$$

where $\Gamma(\ell) \sqsubseteq \{a : Ch(\psi')\}$.
From $\Gamma \vdash^W_\ell k : \psi'$ we deduce that there are some Γ' and ψ such that $\Gamma = \Gamma', k : \psi$ and $\psi \sqsubseteq \psi'$. Therefore, we have $\Gamma', k : \psi, k' : \psi' \vdash_\ell P$ and by the lemma B.1(2), $\Gamma', k : \psi \vdash_\ell [k/k']P$. Then, $\Gamma \vdash_\ell [k/k']P$ is provable.

The other cases of communication simply use B.1(2).

Let us now study the unfolding, that is the case of $(\mathsf{rec}\,A(u).P)(v) \to [\mathsf{rec}\,A(u).P/A][v/u]P$ and assume that the sequent $\Gamma \vdash_\ell (\mathsf{rec}\,A(u).P)(v)$ is provable. We only look at the cases where u and v are simple channels. The proof has the following structure:

$$\dfrac{\dfrac{\dfrac{\vdots}{A : Ch(\gamma), \Delta \vdash_\ell P}}{\Gamma \vdash_\ell \mathsf{rec}\, A(b).P : Ch(\gamma)} \qquad \dfrac{}{\Gamma \vdash_\ell^W c : \gamma}}{\Gamma \vdash_\ell (\mathsf{rec}\, A(b).P)(c)}$$

where $\Gamma(\ell) \sqsubseteq \{c : \gamma\}$, $\Delta =_{dom(\Gamma)-\ell} \Gamma$ and $\Delta(\ell) = [\{b : \gamma, \rho'_{L\cup\{b\}}\}/\rho_L]\Gamma(\ell)$ with $\rho_L = \rho var(\Gamma(\ell))$ and $\rho'_{L\cup\{b\}}$ is fresh. As previously, the sequent $A : Ch(\gamma), \Gamma \vdash_\ell [c/b]P$ is provable. Moreover, since $\Gamma \vdash_\ell \mathsf{rec}\, A(b).P : Ch(\gamma)$ and by the lemma B.1(4), we conclude that $\Gamma \vdash_\ell [\mathsf{rec}\, A(b).P/A][c/b]P$. □

C Unification Proofs

Lemma C.1 *The following propositions hold:*

1. *μ is a solution of $\{\{a : \gamma, \psi\} \doteq \{a : \gamma', \psi'\}\} \cup C$ iff it is a solution of $\{\gamma \doteq \gamma', \psi \doteq \psi'\} \cup C$.*

2. *Let ψ and ψ' be location types such that $chan(\psi) \cap chan(\psi') = \emptyset$, $\rho_L = \rho var(\psi)$ and $\rho'_{L'} = \rho var(\psi')$; μ unifies $\{\psi \doteq \psi'\} \cup C$ iff*

   (a) $\rho_L \notin var(\psi') - \rho'_{L'}$ and $\rho'_{L'} \notin var(\psi) - \rho_L$;

   (b) $chan(\psi) \cap L' = \emptyset$ and $chan(\psi') \cap L = \emptyset$;

   (c) for any $\mathcal{X}'$, for any $\rho''_{L\cup L'} \notin \mathcal{X} \cup \mathcal{X}'$ (where $\mathcal{X}$ is the set of type variables occurring in $\{\psi \doteq \psi'\} \cup C$), there exists λ such that $\mu =_{\mathcal{X}\cup\mathcal{X}'} \lambda\mu'$ and λ unifies $\mu' C$ where $\mu' = [\psi \triangleleft \rho''_{L\cup L'}/\rho'_{L'}\ ,\ \psi' \triangleleft \rho''_{L\cup L'}/\rho_L]$.

Proof:

1. The proof is straightforward.

2. $(\Rightarrow)$ We easily prove (a) reasoning by contradiction that is arising the inexistence of a unifier for ψ and ψ' whenever (a) is not satisfied. For (b), assuming there is some $k \in \{1 \ldots n\}$ such that $a_k \in L'$, there is no substitution substituting a location type ψ'' for $\rho'_{L'}$ with $a_k \in chan(\psi'')$, that is unifying ψ and ψ'. To prove (c), assume $\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ and $\psi' = \{b_1 : \gamma'_1, \ldots, b_m : \gamma'_m, \rho'_{L'}\}$. Since μ unifies ψ and ψ' we have $\{a_1 : \mu\gamma_1, \ldots, a_n : \mu\gamma_n, \mu\rho_L\} = \{b_1 : \mu\gamma'_1, \ldots, b_m : \mu\gamma'_m, \mu\rho'_{L'}\}$, and since $\{a_1, \ldots, a_n\} \cap \{b_1, \ldots, b_m\} = \emptyset$, there is some ψ'' such that

   $$\mu\rho_L = \{b_1 : \mu\gamma'_1, \ldots, b_m : \mu\gamma'_m, \psi''\} = \{a_1 : \mu\gamma_1, \ldots, a_n : \mu\gamma_n, \psi''\} = \mu\rho'_{L'}$$

   Let λ be defined as
   $$\lambda t = \begin{cases} \psi'' & \text{if, } t = \rho''_{L\cup L'} \\ \mu t & \text{otherwise} \end{cases}$$
   Then, since $\rho_L, \rho'_{L'} \notin (var(\psi') \cup var(\psi)) - \{\rho_L, \rho'_{L'}\}$ it is easy to see that $\mu\rho_L = \lambda\mu'\rho_L$ and $\mu\rho'_{L'} = \lambda\mu'\rho'_{L'}$ and $\mu =_{\mathcal{X}\cup\mathcal{X}'} \lambda\mu'$.
   $(\Leftarrow)$ Straightforward. □

Lemma C.2 *Given a problem (C, A), a sequence of reductions*

$$(C\ ,\ A\ ,\ \emptyset) \rightarrow (C_1\ ,\ A_1\ ,\ \mu_1) \rightarrow \ldots$$

terminates either with $\perp$ or with $(\emptyset\ ,\ A'\ ,\ \mu)$ for A' being of atomic subtyping inequations and μ be a substitution.

Proof: Let n_1 and n_2 be defined as follows:

- n_1 is the number of distinct variables occurring in $(C\ ,\ A)$;

- n_2 is the sum of the sizes of the types in $(C\ ,\ A)$ (assuming the size of type variables and constants being of size 1);

Associating the pair $(0, 0)$ to $\perp$, the following table shows the strict decreasing (for the lexicographical order) of (n_1, n_2) for each rule except *(st_wrg)*.

	(trivial$_{(val)}$)/(chan)/(at)/(loc)/(st_ok)	*(loc_end)/(clash)/(oc)/(elim)*
n_1	$=$	$<$
n_2	$<$	

The rule *(st_wrg)* may seems problematic since while n_1 remains unchanged, it may increase n_2. Actually, we can show that this rule can be applied only a finite number of times. The number of assertions in A being constant along the rewriting, it can be ordered as $\psi_0 \dot{\sqsubseteq} \psi_1, \ldots, \psi_{n-1} \dot{\sqsubseteq} \psi_n$. We note that if $(C\ ,\ \{\psi_0 \dot{\sqsubseteq} \psi_1, \ldots, \psi_{n-1} \dot{\sqsubseteq} \psi_n\}\ ,\ \mu) \rightarrow (C'\ ,\ \{\psi'_0 \dot{\sqsubseteq} \psi'_1, \ldots, \psi'_{n-1} \dot{\sqsubseteq} \psi'_n\}\ ,\ \mu')$ then for all i, if $\rho var(\psi_i) = \rho_L$ and $\rho var(\psi'_i) = \rho'_{L'}$ we have $L \subseteq L'$. In particular, if the rule applied is *(st_wrg)* to an assertion $\psi_i \sqsubseteq \psi_{i+1}$, then the inclusion is strict (that is $L \subset L'$). Since we add only names occurring in $(C\ ,\ A\ ,\ \mu)$, such an increasing of L is bounded that proves that we can apply *(st_wrg)* to an assertion only a finite number of times. Then, after a finite number of steps we reach a tuple for which all rules apply except *(st_wrg)* strictly decreasing (n_1, n_2) that is terminating on $(0, 0)$ which corresponds either to $\perp$ or to $(\emptyset\ ,\ A\ ,\ \mu)$. It easy to see that A is necessarily atomic. □

Lemma C.3 *If* $(C\ ,\ A\ ,\ \mu) \rightarrow (C'\ ,\ A'\ ,\ \mu')$ *then*

- *either* $\mu = \mu'$ *and* λ *is a solution for* (C, A) *iff it is a solution for* (C', A')*,*
- *or* $\mu' = \mu''\mu$ *and* $\lambda =_{var(C,A)} \lambda'\mu''$*, for some* λ *and* λ'*, then* λ *is a solution for* (C, A) *iff* λ' *is a solution for* (C', A')*.*

Proof: The case $\mu' = \mu$ corresponds to the rules *(trivail$_{(val)}$)*, *(chan)*, *(at)*, *(loc)* and *(st_ok)* and is easily proved. We prove the other case for the rules *(elim)* and *(st_wrg)* (for *(loc_end)* we simply use lemma C.1).

(elim) $(\Rightarrow)$ Let $\mu'' = [\tau/t]$, and from λ solution of $\{t = \tau\} \cup C$ we deduce that λ has the form

$$[\tau_1/t_1, \ldots, \tau_n/t_n, \lambda\tau/t]$$

that is $\lambda = \lambda'[\tau/t]$ where $\lambda' = [\tau_1/t_1, \ldots, \tau_n/t_n]$. Since λ is solution of $(C\ ,\ A)$, λ' is obviously solution of $([\tau/t]C\ ,\ [\tau/t]A)$.
$(\Leftarrow)$ Straightforward.

(st_wrg) $(\Rightarrow)$ Let λ be a solution of $(C\ ,\ \{\psi_1 \dot{\sqsubseteq} \psi_2\} \cup A)$ and

$$\psi_1 = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}, \quad \psi_2 = \{b_1 : \gamma'_1, \ldots, b_m : \gamma'_m, \rho'_{L'}\}$$

$\rho''_{L \cup L'}$ a fresh row variable and $S'' = [\psi_2 \triangleleft \rho''_{L \cup L'}/\rho_L]$. Since λ is a solution of $\psi_1 \dot{\sqsubseteq} \psi_2$ we have $\lambda\rho_L = \{b_1 : \lambda\gamma'_1, \ldots, b_m : \lambda\gamma'_m, \psi_3\}$ and we deduce that λ has the form

$$[\tau_1/t_1, \ldots, \tau_k/t_k, \{b_1 : \lambda\gamma'_1, \ldots, b_m : \lambda\gamma'_m, \psi_3\}/\rho_L]$$

that is $\lambda = \lambda'\mu''$ where $\lambda' = [\tau_1/t_1, \ldots, \tau_k/t_k, \psi_3/\rho''_{L \cup L'}]$ and it is trivially a solution of $(\mu''C\ ,\ \{\psi_1 \triangleleft \rho''_{L \cup L'} \dot{\sqsubseteq} \rho'_{L'}\} \cup \mu''A)$.
$(\Leftarrow)$ There is no difficulty. □

Proposition C.4 (Soundness) *If* $(C\ ,\ A\ ,\ \emptyset) \rightarrow^* (\emptyset\ ,\ A'\ ,\ \mu)$ *and* $(\emptyset\ ,\ A'\ ,\ \mu)$ *does not reduce anymore, then* μ *is a most general solution w.r.t.* $var(C, A)$ *of* C *and is a solution for* (C, A)*.*

Proof: From the lemma C.2, A' is a set of atomic subtyping assertions. According to lemma C.3, if λ is a solution for (C, A) then there exists λ' solution of $(\emptyset, A')$ such that $\lambda =_{var(C,A)} \lambda'\mu$. Moreover, the identity substitution Id being a solution of $(\emptyset, A')$, $\mu = \mathsf{Id}\mu$ is a solution of (C, A) and we conclude that μ is a mgu w.r.t. $var(C, A)$ for C. □

Lemma C.5 $(C\ ,\ A\ ,\ \emptyset) \rightarrow^* \perp$ *iff* (C, A) *has no solution.*

Proof: $(\Rightarrow)$ It is straightforward.
$(\Leftarrow)$ By contradiction using proposition C.4 and lemma C.2. □

Proposition C.6 (Completeness) *If* (C, A) *has a solution, then* $(C\ ,\ A\ ,\ \emptyset) \rightarrow^* (\emptyset\ ,\ A'\ ,\ \mu)$*.*

Proof: It is corollary of lemmas C.5 and C.1. □

D Type Inference Proofs

Lemma D.1 *Let u be a (possibly compound) name and t be a type variable, $\ell \in \mathcal{N}_{loc}$, and Γ, C, A such that $gen(u\colon t, \ell) = (\Gamma, C, A)$, then*

1. *for all solution μ of (C, A), we have $\mu\Gamma \vdash_\ell u\colon \mu t$.*
2. *for all Γ', τ such that $\Gamma' \vdash_\ell u\colon \tau$, there exists a solution μ of (C, A) such that $\Gamma' = \mu\Gamma$ and $\tau = \mu t$.*

Lemma D.2 *Given $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$, a sequence of reductions*

$$(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A) \rightarrowtriangle (J'_s\,,\ J'_p\,,\ J'_n\,,\ C'\,,\ A') \rightarrowtriangle \ldots$$

terminates either on $\perp$ or on $(\emptyset\,,\ \emptyset\,,\ \emptyset\,,\ C''\,,\ A'')$.

Proof Sketch: We define

- n_1 : the sum of sizes of terms in J_s;
- n_2 : the sum of sizes of terms in J_p;
- n_3 : the number of sequents in J_n;

Associating the triple (n_1, n_2, n_3) to each tuple $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$, and $(0, 0, 0)$ to $\perp$, it is easy to see that this triple is strictly decreasing along the reduction steps for the lexicographical order. This shows the terminaison of $\rightarrowtriangle$. □

Lemma D.3 (Preservation) *Let μ solution of $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$ and λ of $(J'_s\,,\ J'_p\,,\ J'_n\,,\ C'\,,\ A')$, if $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A) \rightarrowtriangle (J'_s\,,\ J'_p\,,\ J'_n\,,\ C'\,,\ A')$ then*

1. *there is $\lambda' =_{\mathcal{X}} \mu$ solution of $(J'_s\,,\ J'_p\,,\ J'_n\,,\ C'\,,\ A')$, and $\mathcal{X} = var(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$,*
2. *λ is also solution of $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$.*

Proof: (1) We prove the lemma for some relevant rules. Let μ be a solution of $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$.

(n₁) Let $J_n = \{\Gamma \vdash^W_\ell \ell'\colon \psi\} \cup J'_n$ then in particular μ is solution of $\Gamma \vdash^W_\ell \ell'\colon \psi$, that is $\mu\Gamma \vdash^W_\ell \ell'\colon \mu\psi$ and $\mu\Gamma(\ell') \sqsubseteq \mu\psi$; then $\lambda' = \mu$ is a solution of $(J_s\,,\ J_p\,,\ J'_n\,,\ C\,,\ A \cup \{\Gamma(\ell') \mathrel{\dot{\sqsubseteq}} \psi\})$.

(n₂) Let $J_n = \{\Gamma \vdash^W_\ell a\colon \gamma\} \cup J'_n$, then in particular μ is solution of $\Gamma \vdash^W_\ell a\colon \gamma$, that is $\mu\Gamma \vdash^W_\ell a\colon \mu\gamma$ and $\mu\Gamma(\ell) \sqsubseteq \{a\colon \mu\gamma, \rho'_{\{a\}}\}$. From this assertion we deduce that $\mu\Gamma(\ell) = \{a\colon \mu\gamma, \psi\}$ for some ψ. Let λ' be defined as μ on $\mathcal{X}$ and $\lambda'\rho_{\{a\}} = \psi$, then λ' is obviously a solution of $(J_s\,,\ J_p\,,\ J'_n\,,\ C \cup \{\Gamma(\ell) \doteq \{a\colon \gamma, \rho_{\{a\}}\}\}\,,\ A)$.

(n₃) Let $J_n = \{\Gamma \vdash^W_\ell a@\ell\colon \tau\} \cup J'_n$, then μ solution of $\Gamma \vdash^W_\ell a@\ell\colon \tau$ implies that there are some γ and ψ such that $\mu\tau = \gamma@\psi$ and $\mu\Gamma(\ell) \sqsubseteq \{a\colon \gamma, \psi\}$. Therefore, there is also some ψ' such that $\mu\Gamma(\ell) = \{a\colon \gamma, \psi'\}$ and $\psi' \sqsubseteq \psi$. We set λ' such that $\lambda' =_{\mathcal{X}} \mu$ and $\lambda' h = \gamma$, $\lambda'\rho'_\emptyset = \psi$ and $\lambda'\rho_{\{a\}} = \psi'$. Then, we easily see that λ' is solution of $(J_s\,,\ J_p\,,\ J'_n\,,\ C \cup \{\Gamma(\ell) \doteq \{a\colon h, \rho_{\{a\}}\}, h@\rho'_\emptyset \doteq \tau\}\,,\ \{\rho_{\{a\}} \mathrel{\dot{\sqsubseteq}} \rho'_\emptyset\} \cup A)$.

(p₁) Let $J_p = \{\Gamma \vdash_\ell \overline{a}u\} \cup J'_p$, μ solution of $\Gamma \vdash_\ell \overline{a}u$ means $\mu\Gamma \vdash_\ell \overline{a}u$, that is $\mu\Gamma \vdash^W_\ell u\colon \mu\tau$ and $\mu\Gamma(\ell) \sqsubseteq \{a\colon \mathit{Ch}(\mu\tau), \rho_{\{a\}}\}$. Then, there is some ψ such that $\mu\Gamma(\ell) = \{a\colon \mathit{Ch}(\mu\tau), \psi\}$. Therefore, defining λ' as μ on $\mathcal{X}$, $\lambda' t = \lambda'\tau$ and $\lambda'\rho''_{\{a\}} = \psi$, This substitution is solution of $(J_s\,,\ J'_p\,,\ J_n \cup \{\Gamma \vdash^W_\ell u\colon t\}\,,\ C \cup \{\Gamma(\ell) \doteq \{a\colon \mathit{Ch}(t), \rho''_{\{a\}}\}\}\,,\ A)$.

(p₂) Let $J_p = \{\Gamma \vdash_\ell a(u).P\} \cup J'_p$, we have $\mu\Gamma \vdash_\ell a(u).P$.

- if $u = b$ then $\mu\Gamma(\ell) = \{a\colon \mathit{Ch}(\gamma), \psi\}$ and $\Delta' \vdash_\ell P$ where Δ' is defined as

$$\Delta' =_{\mathit{dom}(\Gamma)-\ell} \mu\Gamma \;\text{ and }\; \Delta'(\ell) = [\{b : \gamma; \rho'''_{L'\cup\{b\}}\}/\rho''_{L'}]\mu\Gamma(\ell)$$

  for some γ and ψ, and where $\rho''_{L'} = \rho\mathit{var}(\mu\Gamma(\ell))$. Defining λ' as μ on $\mathcal{X}$, $\lambda' t = \lambda' h = \gamma$, $\lambda'\rho^4_{\{a\}} = \psi$ and $\lambda'\rho_L = \lambda'\rho'_{L\cup\{b\}} = [\rho'''_{L'\cup\{b\}}/\rho''_{L'}]\mu\rho_L$, it is easy to show that $\Delta' = \lambda'\Delta$, then that λ' is solution of

$$(J_s\,,\ \{\Delta \vdash_\ell P\} \cup J'_p\,,\ J_n\,,\ C \cup \{t \doteq h, \rho_L \doteq \rho'_{L\cup\{b\}}, \Gamma(\ell) \doteq \{a\colon \mathit{Ch}(t), \rho^4_{\{a\}}\}\}\,,\ A)$$

- if $u \notin \mathcal{N}_{chan}$, then $\mu\Gamma, \Delta \vdash_\ell P$ and $\Delta \vdash_\ell u : \tau$. Let $(\Gamma', C'') = gen(u : t, \ell)$, from the lemma D.1 there exists a substitution λ solution of C'' such that $\Delta = \lambda\Gamma'$ and $\tau = \lambda t$. Since we can choose Γ' such that $var(\Gamma) \cap var(\Gamma') = \emptyset$, we can define λ' as μ on $\mathcal{X}$ and as λ on $var(C'')$.

(p_{4b}) We define λ' as μ on $\mathcal{X}$ and $\lambda'\rho_\emptyset = \mu\Gamma(a)$ and $\lambda'\rho'_\emptyset = \mu\Gamma(b)$.

The remaining cases are trivial or are quite similar to *(p_1)* and *(p_2)*.

(2) We prove this point for only two relevant rules, the other cases being either trivial or similar to those ones. Let λ be a solution of $(J'_s\,,\, J'_p\,,\, J'_n\,,\, C'\,,\, A')$. The proof for *($n_1$)* to *(nc)* are straightforward.

(p_1) We have $J'_n = J_n \cup \{\Gamma \vdash_\ell u : t\}$ and $C' = C \cup \{\Gamma(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}\}$ then $\lambda\Gamma \vdash^W_\ell u : \lambda t$ and $\lambda\Gamma(\ell) \sqsubseteq \{a : Ch(\lambda t), \rho'_{\{a\}}\}$ for some fresh $\rho'_{\{a\}}$. According to the inference rule for the output λ is therefore a solution of $\Gamma \vdash_\ell \overline{a}u$.

(p_2) We have $J'_p = \{\Delta \vdash_\ell P\} \cup J''_p$ and $C' = \{\Gamma(\ell) \doteq \{a : Ch(t), \rho''_{\{a\}}\}\} \cup C'' \cup C$.

- Considering the case where $u = b$, $C'' = \{t \doteq h, \rho_L \doteq \rho'_{L\cup\{b\}}\}$ and Δ is defined as

$$\Delta =_{dom(\Gamma)-\ell} \Gamma \ \text{ and } \ \Delta(\ell) = [\{b : \gamma, \rho'_{L\cup\{b\}}\}/\rho_L]\Gamma(\ell)$$

  and $\rho_L = \rho var(\Gamma(\ell))$. $\Delta(\ell)$ is of the form $\{b : h, a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho'_{L\cup\{b\}}\}$ if

$$\Gamma(\ell) = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$$

  and

$$\lambda\Delta(\ell) = \{b : \lambda h, a_1 : \lambda\gamma_1, \ldots, a_n : \lambda\gamma_n, b_1 : \gamma'_1, \ldots, b_m : \gamma'_m, \rho'''_{L'\cup\{b\}}\}$$

  if $\lambda\rho'_{L\cup\{b\}} = \{b_1 : \gamma'_1, \ldots, b_m : \gamma'_m, \rho'''_{L'\cup\{b\}}\} = \lambda\rho_L$ (because λ unifies ρ_L and $\rho_{L\cup\{b\}}$). Then we have

$$\lambda\Gamma(\ell) = \{a_1 : \lambda\gamma_1, \ldots, a_n : \lambda\gamma_n, b_1 : \gamma'_1, \ldots, b_m : \gamma'_m, \rho'''_{L'\cup\{b\}}\}$$

  then, since $b \notin nm(\lambda\Gamma)$,

$$[\{b : \lambda h, \rho^4_{L'\cup\{b\}}\}/\rho'''_{L'\cup\{b\}}]\lambda\Gamma(\ell) = [\rho^4_{L'\cup\{b\}}/\rho'''_{L'\cup\{b\}}]\lambda\Delta(\ell)$$

  Also

$$[\rho^4_{L'\cup\{b\}}/\rho'''_{L'\cup\{b\}}]\lambda\Gamma =_{dom(\Gamma)-\ell} [\rho^4_{L'\cup\{b\}}/\rho'''_{L'\cup\{b\}}]\lambda\Delta$$

  We can easily see that, since λ is a solution of $\Delta \vdash_\ell P$, $[\rho^4_{L'\cup\{b\}}/\rho'''_{L'\cup\{b\}}]\lambda$ is also a solution of that sequent. Therefore $\lambda\Gamma \vdash_\ell a(b).P$.

- In the other case $\Delta = \Gamma, \Gamma'$, $A' = A \cup A''$ and $gen(u : t, \ell) = (\Gamma', C'', A'')$. From the lemma D.1 and since λ is solution of (C'', A''), we have $\lambda\Gamma' \vdash_\ell u : \lambda t$ and we can see that $\lambda\Gamma, \lambda\Gamma' \vdash_\ell P$ therefore λ is a solution of $\Gamma \vdash_\ell a(u).P$. □

Lemma D.4 *Let S be a typable network, $N = fn(S) \cap (\mathcal{N}_{val} \cup \mathcal{N}_{loc})$ and Γ a context such that $\Gamma \vdash S$ then $N \subset dom(\Gamma)$ and $\Gamma_{/N} \vdash S$ (where $\Gamma_{/N}$ is the restriction of Γ to the domain N).*

Theorem D.5 (Soundness) *Let S be a network and Γ be an initial context for S, if*

$$(\{\Gamma \vdash S\}\,,\, \emptyset\,,\, \emptyset\,,\, \emptyset\,,\, \emptyset) \twoheadrightarrow^* (\emptyset\,,\, \emptyset\,,\, \emptyset\,,\, C\,,\, A) \ \text{ and } (C\,,\, A\,,\, \emptyset) \rightarrow^* (\emptyset\,,\, A'\,,\, \mu)$$

then $\mu\Gamma; A'$ is a principal typing for S.

Proof:

1. We first have to show that for all A'-substitution λ, $\lambda\mu\Gamma \vdash S$. Since λ is a A'-substitution, it is a solution for $(\emptyset, A')$ and, from the lemma C.3, it is easy to see that $\lambda\mu$ is a solution for (C, A). Then, by the previous lemma, $\lambda\mu$ is a solution for $\Gamma \vdash S$.

2. Secondly, let Γ' be a context such that $\Gamma' \vdash S$. From the lemma D.4, we have $\Gamma'_{/N} \vdash \mu S$ and it is easy to see that there exists a substitution μ' such that $\Gamma'_{/N} = \mu'\Gamma$. We can assume, without loss of generality, that $dom(\mu') \subseteq var(\Gamma)$. Let λ be defined as follows

$$\lambda t = \begin{cases} \mu' t & \text{if, } t \in var(\Gamma) \\ \mu t & \text{otherwise.} \end{cases}$$

From the assumptions, we can deduce that $\mu'\Gamma = \lambda\Gamma$, then $\lambda\Gamma \vdash S$, that is λ is a solution of $\Gamma \vdash S$. Therefore, from the lemma D.3, there exists a substitution $\lambda' =_{var(\Gamma)} \lambda$ solution of (C, A). Since μ is a most general solution w.r.t. $var(C, A)$, there exists some A'-substitution λ'' such that $\lambda' =_{var(C,A)} \lambda''\mu$. Moreover, since $\lambda' =_{var(\Gamma)} \mu'$ and $var(\Gamma) \subseteq var(C, A)$, we deduce $\mu' =_{var(C,A)} \lambda''\mu$. Therefore, $\Gamma' =_{dom(\Gamma)} \lambda''\mu\Gamma$. □

Lemma D.6 *Let $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A)$ be a tuple that has a solution, then $(J_s\,,\ J_p\,,\ J_n\,,\ C\,,\ A) \not\twoheadrightarrow^* \bot$.*

Proof: By the lemma D.2 we know that there is no infinite sequence of reductions. Then, the proof is simply by induction on the length of the sequence of reductions. □

Theorem D.7 (Completeness) *Let S be typable network and Γ be an initial context for S', then*

$$(\{\Gamma \vdash S\}\,,\ \emptyset\,,\ \emptyset\,,\ \emptyset\,,\ \emptyset) \twoheadrightarrow^* (\emptyset\,,\ \emptyset\,,\ \emptyset\,,\ C\,,\ A) \ \textit{ and } \ (C\,,\ A\,,\ \emptyset) \rightarrow^* (\emptyset\,,\ A'\,,\ \mu)$$

Proof: Since S is typable, there is some Γ' such that $\Gamma' \vdash S$. By the lemma D.4, we know that $\Gamma'_{/N} \vdash S$ (where $N = fn(S) \cap (\mathcal{N}_{val} \cup \mathcal{N}_{loc})$). Moreover, following the assumptions it is easy to define a substitution μ' such that $\mu'\Gamma = \Gamma'_{/N}$, that is μ' is a solution of $\Gamma \vdash S$. Then, by the lemmas D.2 and D.6 there exist some C and A such that $(\{\Gamma \vdash S\}\,,\ \emptyset\,,\ \emptyset\,,\ \emptyset\,,\ \emptyset) \twoheadrightarrow^* (\emptyset\,,\ \emptyset\,,\ \emptyset\,,\ C\,,\ A)$ and, by D.3, a substitution $\mu'' =_{var(\Gamma)} \mu'$ solution of (C, A). By the lemmas C.4 and C.6, $(C\,,\ A\,,\ \emptyset) \rightarrow^* (\emptyset\,,\ A'\,,\ \mu)$ and there exists a substitution λ such that $\mu'' =_{var(C,A)} \lambda\mu$. Therefore, $\mu' =_{var(\Gamma)} \lambda\mu$. □

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifi que
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399